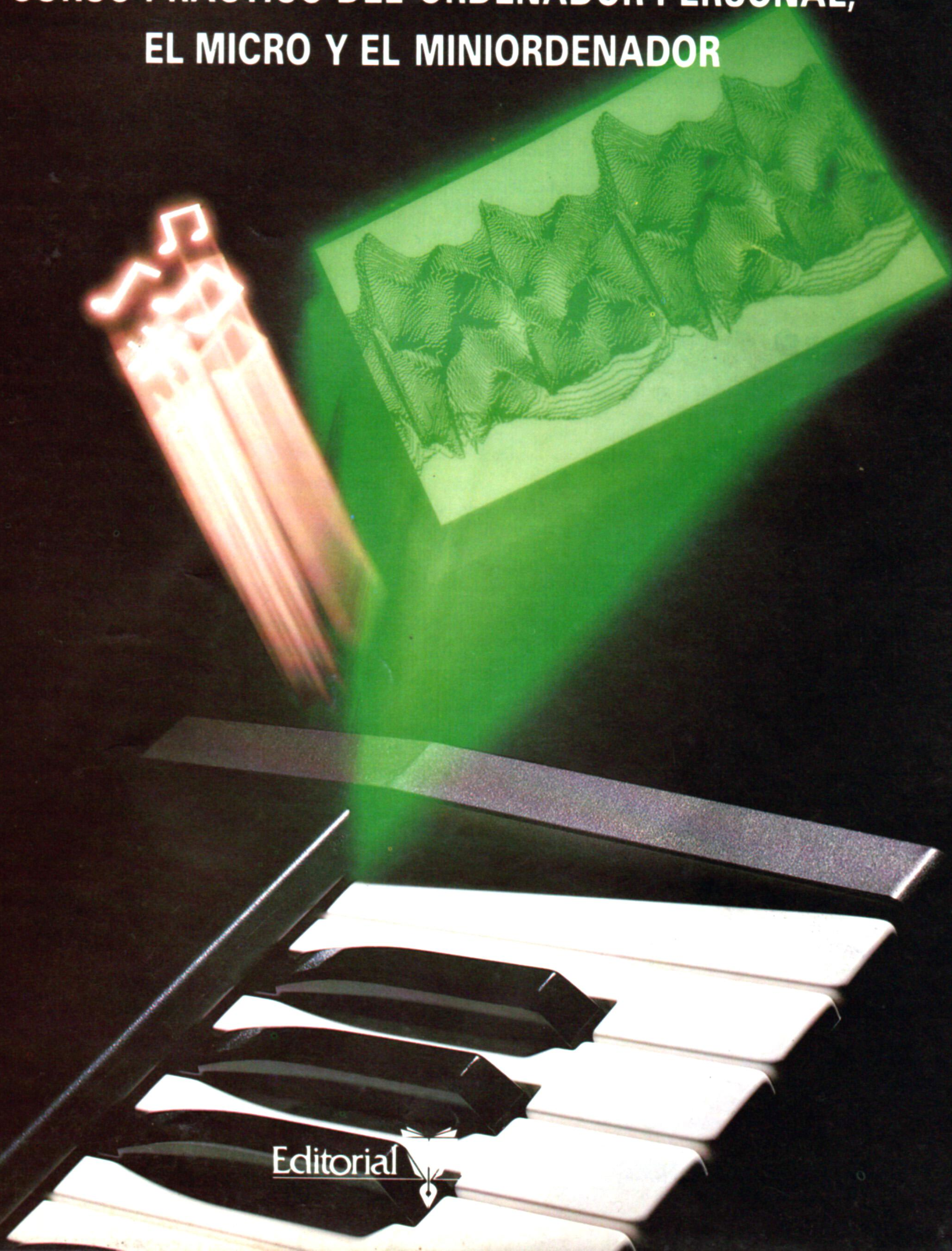
175 PTAS

92

MI COMPUTER

CURSO PRACTICO DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR



Editorial

MI COMPUTER CURSO PRACTICO DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR

Publicado por Editorial Delta, S.A., Barcelona

Volumen VIII-Fascículo 92

Director: José Mas Godayol
Director editorial: Gerardo Romero
Jefe de redacción: Pablo Parra
Coordinación editorial: Jaime Mardones, Francisco Martín
Asesor técnico: Ramón Cervelló

Redactores y colaboradores: G. Jefferson, R. Ford, F. Martín, S. Tarditti, A. Cuevas, F. Blasco
Para la edición inglesa: R. Pawson (editor), D. Tebbutt (consultant editor), C. Cooper (executive editor), D. Whelan (art editor), Bunch Partworks Ltd. (proyecto y realización)

Realización gráfica: Luis F. Balaguer

Redacción y administración:
Aribau, 185, 1.°, 08021 Barcelona
Tel. (93) 209 80 22

MI COMPUTER, *Curso práctico del ordenador personal, el micro y el miniordenador*, se publica en forma de 96 fascículos de aparición semanal, encuadernables en ocho volúmenes. Cada fascículo consta de 20 páginas interiores y sus correspondientes cubiertas. Con el fascículo que completa cada uno de los volúmenes, se ponen a la venta las tapas para su encuadernación.
El editor se reserva el derecho de modificar el precio de venta del fascículo en el transcurso de la obra, si las circunstancias del mercado así lo exigieran.

ISBN: 84-85822-83-8 (fascículo) 84-7598-067-2 (tomo 7)
84-85822-82-X (obra completa)
Depósito Legal: B. 52-84

Fotocomposición: Tecfa, S.A., Pedro IV, 160, Barcelona-5
Impresión: Cayfosa, Santa Perpètua de Mogoda (Barcelona) 308510
Impreso en España-Printed in Spain-Octubre 1985

Editorial Delta, S.A., garantiza la publicación de todos los fascículos que componen esta obra.

Distribuye para España: Marco Ibérica, Distribución de Ediciones, S.A., Carretera de Irún, km 13,350. Variante de Fuencarral, 28034 Madrid.
Distribuye para Colombia: Distribuidoras Unidas, Ltda., Transversal 93, n.° 52-03, Bogotá D.E.
Distribuye para México: Distribuidora Intermex, S.A., Lucio blanco, n.° 435, Col. San Juan Tlihuaca, Azcapotzalco, 02400, México D.F.
Distribuye para Venezuela: Distribuidora Continental, S.A., Edificio Bloque Dearmas, final Avda. San Martín con final Avda. La Paz, Caracas 1010.

Pida a su proveedor habitual que le reserve un ejemplar de MI COMPUTER. Comprando su fascículo todas las semanas y en el mismo quiosco o librería, Vd. conseguirá un servicio más rápido, pues nos permite realizar la distribución a los puntos de venta con la mayor precisión.

Servicio de suscripciones y atrasados (sólo para España)

Las condiciones de suscripción a la obra completa (96 fascículos más las tapas, guardas y transferibles para la confección de los 8 volúmenes) son las siguientes:

a) Un pago único anticipado de 19 425 ptas. o bien 8 pagos trimestrales anticipados y consecutivos de 2 429 ptas. (sin gastos de envío).

b) Los pagos pueden hacerse efectivos mediante ingreso en la cuenta 6.850.277 de la Caja Postal de Ahorros y remitiendo a continuación el resguardo o su fotocopia a Editorial Delta, S. A. (Aribau, 185, 1.°, 08021 Barcelona), o también con talón bancario remitido a la misma dirección.

c) Se realizará un envío cada 12 semanas, compuesto de 12 fascículos y las tapas para encuadernarlos.

Los fascículos atrasados pueden adquirirse en el quiosco o librería habitual. También pueden recibirse por correo, con incremento del coste de envío, haciendo llegar su importe a Editorial Delta, S.A., en la forma establecida en el apartado b).

No se efectúan envíos contra reembolso.

Editorial Delta, S.A.



Centro musical



Paul Chave

Teclado avanzado
El teclado Echo, de LVL, que vemos en la ilustración, es el primer teclado musical "verdadero" producido para el BBC Micro y el Commodore 64 que accede directamente a los chips de sonido de estos ordenadores y que no exige una costosa interface

El paquete de música Echo, destinado al Commodore 64 y al BBC Micro, accede directamente al chip de sonido del micro

El paquete Echo, de Leasalink Viewdata Ltd (LVL), consta de un auténtico teclado de tres octavas y media más el software para hacerlo funcionar. También se proporciona un manual y un cable interface de 20 vías para conectar el teclado y el ordenador entre ellos mismos, que se instala en la puerta para el usuario de la máquina. Si el usuario posee un Commodore 64, se le entregarán dos cables: el cable estándar, que es para el BBC Micro, y un pequeño cable adaptador que se ofrece para producir la interface correcta para la máquina de Commodore.

El hardware del Echo es simple desde el punto de vista de diseño; la mayor parte del trabajo lo realiza el software que viene con el paquete que, obviamente, mantiene reducidos los costos de producción. El software para ambos ordenadores se basa en las mismas especificaciones.

En el software hay disponibles dos modalidades distintas, la primera de las cuales es una modalidad de órgano. Pulsando una de las teclas del teclado del ordenador se seleccionará uno de los diversos "instrumentos" listados. Éstos van desde la guitarra hawaiana hasta el cello y el clavicordio, aunque los fondos de cada una de las dos máquinas son ligeramente diferentes. En la parte superior del monitor se listará una cantidad de parámetros que, según cuál seleccione usted, le permitirán alterar los tonos del instrumento que esté tocando.

La versión del software para el Commodore 64 posee la facilidad de añadir trémolo y vibrato (pero no ambos) o cambiar a una clave mayor o menor. Estos efectos se producen pulsando una de las teclas numéricas.

El software de la versión para el BBC Micro también incorpora las teclas del ordenador, pero en este caso son las teclas de función las que le permitirán establecer el trémolo y acordes mayores o menores. Los programadores han sacado partido de las teclas de función extras y de la velocidad del procesador de la máquina, por lo que han añadido otras facilidades. Si el usuario desea dar un énfasis o un efecto extra cuando ejecuta una pieza en el teclado Echo, puede pulsar una de las teclas de función y se tocará el sonido apropiado (bajo, cimbal, etc.). Lamentablemente, no hay ninguna facilidad que permita tocar ritmos de acompañamiento mientras se esté tocando el teclado.

Controles de altura

En la parte inferior de cada pantalla están los controles de altura. En la versión para el BBC Micro, se puede subir o bajar la altura pulsando las teclas del cursor adecuadas. Sin embargo, en la versión para el Commodore 64 esta labor se realiza mediante las teclas < y >. Lamentablemente, el hecho de que estas dos funciones aparezcan en la pantalla etiquetadas como PITCH MINUS no dice mucho en favor de la calidad del software para la máquina Commodore.

También es lamentable que los sonidos producidos por las teclas de "instrumentos" guarden muy poco parecido con los instrumentos que se pretende imitar. Éste es un hecho harto frecuente en todo el campo de la música electrónica, no sólo en los programados para ordenador. No obstante, los efectos de sonido del Echo parecen cualitativamente peores que la mayoría. Por ejemplo, en la versión para el Commodore 64 el violín, la viola y el cello suenan meramente como un órgano sostenido a diferentes alturas para cada instrumento.

Ambas máquinas disponen, asimismo, de la modalidad de sintetizador. Los parámetros proporcionados por la modalidad de sintetizador dependen de las facilidades que ofrece el chip de sonido de cada máquina. De este modo, la versión para el Commodore 64 permite regular los parámetros de envoltura de la forma de onda y programar el filtro. Tal como sucede en la modalidad de órgano, los parámetros se alteran pulsando la tecla adecuada en el teclado del ordenador hasta obtener el valor requerido.

De una forma bastante similar, la modalidad de sintetizador del BBC Micro hace uso de la instruc-

ECHO SYNTHESISER

TECLADO

37 teclas de recorrido total

INTERFACES

Cable de 20 vías que se enchufa en la puerta para el usuario del BBC Micro. Cable adaptador para el C64

SOFTWARE

La versión para el BBC Micro se vende en cassette o en disco; la versión para el C64 sólo en cassette

VENTAJAS

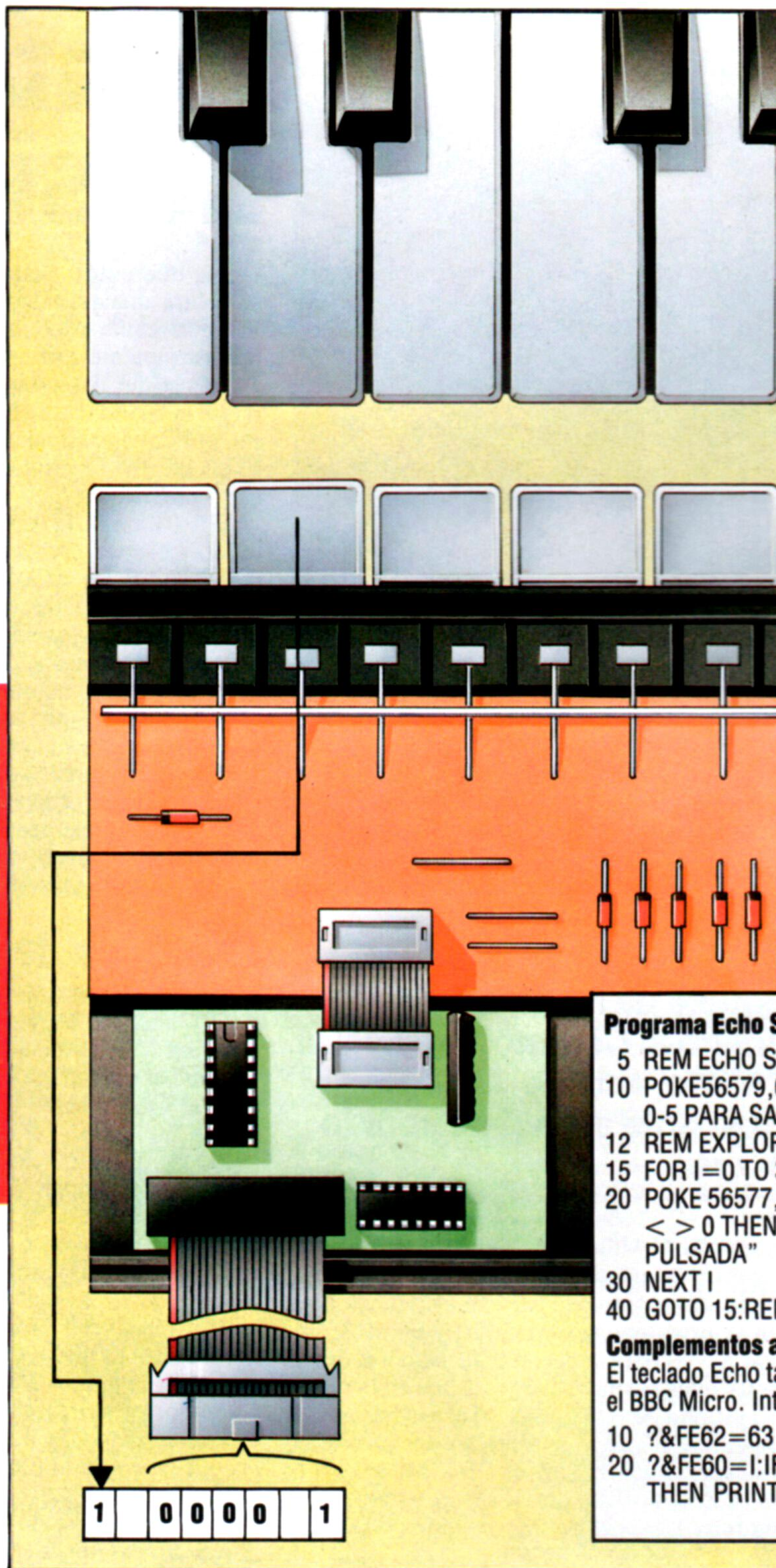
Permite utilizar, desde un auténtico teclado, las facilidades de los chips de sonido de ambos micros

DESVENTAJAS

La pobre calidad del software reduce sus posibilidades, si bien existe la promesa de una versión mejorada



Música en pantalla
Éstas son las visualizaciones en pantalla para el software del Commodore 64 (arriba) en modalidad de órgano, y para el BBC Micro en modalidad de sintetizador. A pesar de ser superficialmente parecidas, las versiones del software son bien diferentes. Aunque ambas utilizan pulsaciones de teclas desde el ordenador para alterar los sonidos, la versión para el BBC Micro posee una variedad de fondos muchísimo mayor que la versión para el Commodore 64



Echo Scan

El teclado Echo se enchufa en la puerta para el usuario del BBC Micro o del Commodore 64 y se lo puede integrar fácilmente desde software. El principio de operación es muy simple. Los seis bits inferiores del registro de datos de la puerta para el usuario se establecen en salida para poder enviar datos al teclado, y el bit más significativo (el 7) se establece en entrada para volver a recibir datos desde el teclado. Para comprobar una pulsación de tecla es preciso, en primer lugar, colocar (POKE) un número entre 0 y 36 en el registro de datos de la puerta para el usuario (el teclado Echo posee 37 teclas numeradas a partir de cero desde el extremo izquierdo). Éste define a la tecla que deseamos comprobar. Luego debemos comprobar el bit 7 del registro de datos (operando el valor del registro con 128 mediante AND). Si este bit es uno, entonces en ese momento se está pulsando la tecla en cuestión; si el valor del bit es cero, ello indica que no se está pulsando la tecla. Para explorar todo el teclado necesitamos, por lo tanto, un bucle que compruebe cada una de las 37 teclas de forma sucesiva. El programa que ofrecemos muestra cómo se puede hacer esto desde BASIC.

Programa Echo Scan

```
 5 REM ECHO SCAN CBM 64
10 POKE56579,63:REM ESTABLECER BITS
   0-5 PARA SALIDA
12 REM EXPLORAR TECLADO DE 37 TECLAS
15 FOR I=0 TO 36
20 POKE 56577,I:IF(PEEK(56577)AND 128)
   < > 0 THEN PRINT I; "TECLA
   PULSADA"
30 NEXT I
40 GOTO 15:REM REPETIR
```

Complementos al BASIC

El teclado Echo también se puede utilizar con el BBC Micro. Introduzca estos cambios:

```
10 ?&FE62=63
20 ?&FE60=I:IF(?FE60 AND 128) < > 0
   THEN PRINT I; "TECLA PULSADA"
```

Kevin Jones

ción ENVELOPE de la máquina. Cuando llame a esta instrucción, se le preguntará cuál de los cuatro sintetizadores definidos por el usuario (en contraposición a aquellos que ya están preestablecidos en el órgano) desea alterar. La pantalla visualiza entonces los 14 parámetros que se requieren para la instrucción ENVELOPE; cada uno de ellos se selecciona mediante las teclas del cursor izquierda y derecha, y sus valores se pueden modificar mediante los cursores arriba y abajo. Una vez establecidos, usted estará en libertad de tocar el tono deseado en el teclado.

El software conecta en interface el ordenador con el teclado colocando un valor de entre 36 y cero en los seis bits inferiores del registro de datos de la puerta para el usuario. El método de exploración de teclado que utiliza el paquete de software significa que si se mantienen pulsadas varias teclas simultáneamente, sólo se registrará la más elevada. Lamentablemente, esto no permite cambios rápidos de dedos en el teclado del Echo, porque antes de que el software reconozca una pulsación de tecla, debe haberse liberado completamente la tecla anterior.



Sonidos de circuito

Prosiguiendo con nuestro proyecto, diseñaremos una interface para el Commodore 64 y el BBC Micro

El principal componente de la interface es un adaptador de interface para comunicaciones asincrónicas (ACIA: *Asynchronous communications interface adaptor*). La función del ACIA consiste en convertir los datos en paralelo del ordenador a la forma en serie que requiere la MIDI, y viceversa. El dispositivo se halla contenido en un circuito integrado. Nosotros emplearemos el MC6850, presente en muchos ordenadores personales como un control para la transferencia asíncrona de datos desde y hacia cassette e interfaces de tipo RS232.

Lista de componentes

Cantidad	Ref.	Artículo
1	C4	Condensador policarbonato 1 nF
2	C2/C3	Condensador policarbonato 100 nF
1	R1	Resistencia 270 ohmios
3	R2/R3/R4	Resistencia 220 ohmios
2	R5/R6	Resistencia 680 ohmios
1		Carrete de alambre para enlaces
2	SK1/SK2	Conector DIN 5 patillas 180° para montaje en PCB
1	IC1	Chip ACIA MC68B50
1	IC2	Chip optoaislante 6N139
1	IC3	Inversor hexa 74LS04 TTL
1		Conector DIL 24 patillas
1		Conector DIL 8 patillas
1		Conector DIL 14 patillas
1	X1	Cristal 2.0 MHz
1	D1	Diodo 1N914

La placa DIP que se utiliza para montar los componentes se puede conseguir en casi todas las sucursales Tandy, bajo el número de componente 276-164

Conexiones del circuito
El optoacoplamiento se consigue con un optoaislante 6N139 (IC2). El reloj de transmisión es derivado del reloj del ordenador mediante un circuito oscilador de cristal de 2MHz formado alrededor de IC3a e IC3b. IC3c, IC3d e IC3e proporcionan cada uno funciones de tamponamiento para el transmisor y generador del reloj. El MC6850 es bus compatible con la CPU 6502 utilizada en muchos ordenadores. Ello nos permite efectuar conexiones directas desde las líneas Read/Write, IRQ y Clock de micros basados en el 6502. (Observe que, para evitar posibles confusiones con las entradas del reloj de transmisión TXD y TXC, hemos etiquetado la entrada del reloj al ACIA como Enable (E)). Las dos líneas de selección del chip (CS0 y CS1) están conectadas a la alimentación de + 5 V y, por tanto, fijadas en *high*, dejando a CS2 como la única entrada activa de selección de chip (o habilitador de dispositivo). Esta línea queda *low* para una cierta gama de direcciones, permitiendo que las operaciones de lectura y escritura en el bus de datos accedan a los registros internos del ACIA

Diagrama de circuitos de la interface MIDI

+ 5 V → + 5 V
TIERRA → 0 V
CONECTOR AL BUS DEL ORDENADOR: D0, D1, D2, D3, D4, D5, D6, D7, A0, CS, R/W, Ø, IRQ
(IC1) MC68B50
IC2 6N139
IC3a 74LS04, IC3b, IC3c, IC3d, IC3e
C2 100 nF; 100nF C3; C4 1nF
R1 270 ohmios; R2 220 ohmios; R3 220 ohmios; R4 220 ohmios; R5 680 ohmios; R6 680 ohmios
D1 1N914
SK1 MIDI OUT (vista exterior)
SK2 MIDI IN (vista exterior)
X1 cristal de 2 MHz

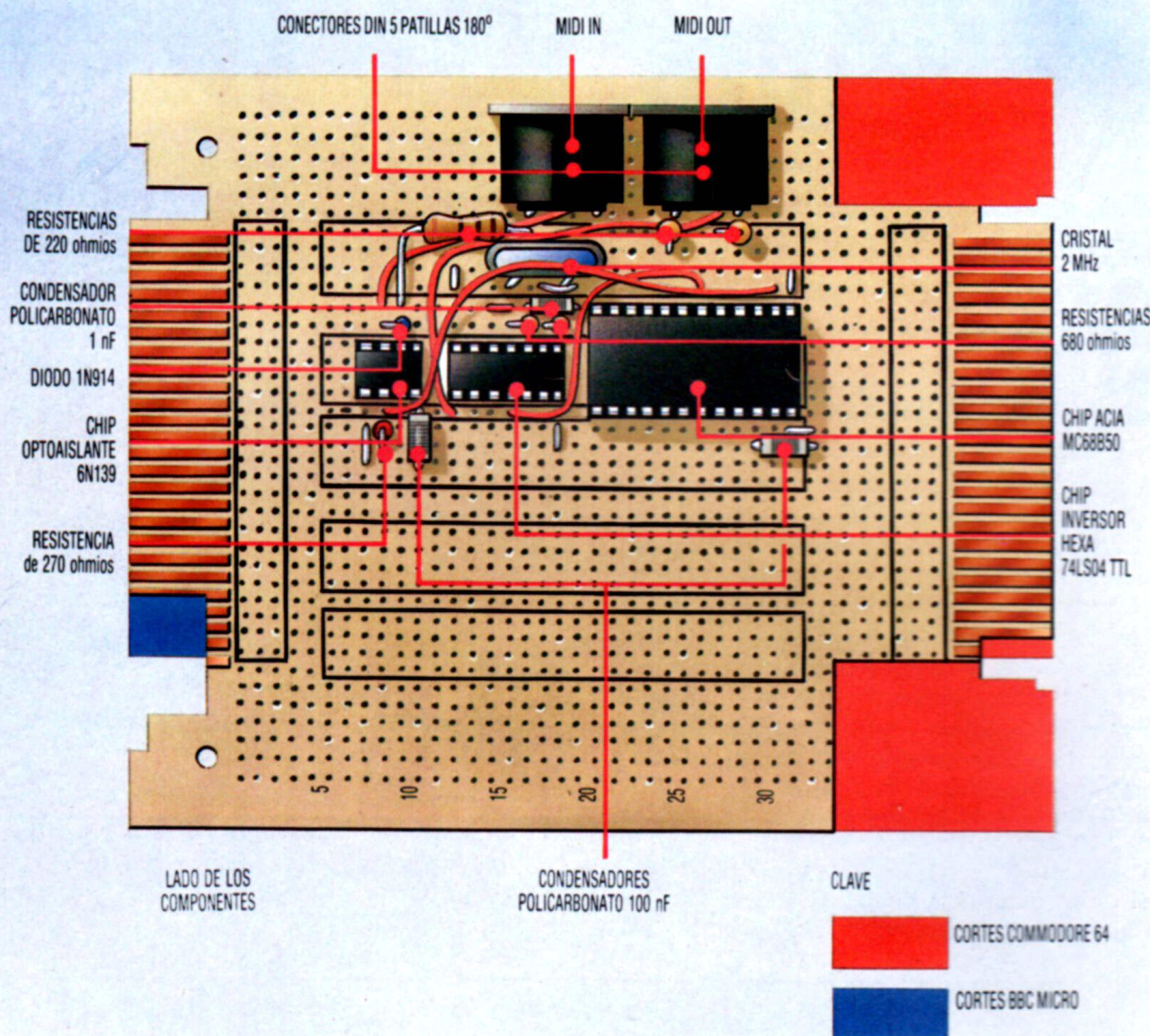
Kevin Jones

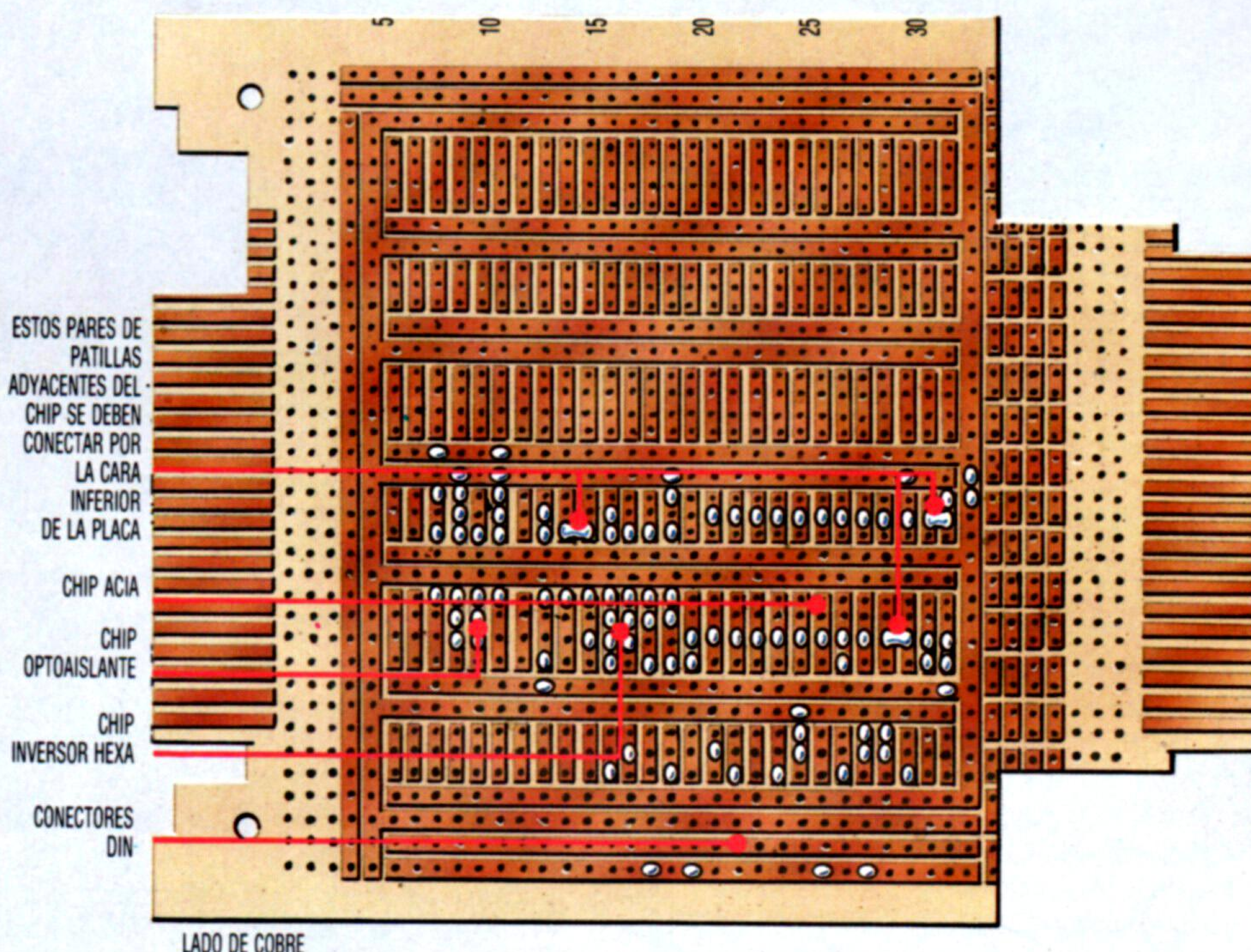
Construcción de la MIDI
Los componentes principales de la interface MIDI se deben montar en la placa DIP especial que se especifica en la lista de componentes. Esta placa posee conectores marginales de doble cara en ambos extremos; el extremo derecho se utiliza para la conexión al Commodore 64, y el extremo izquierdo para la conexión al BBC Micro. Antes de montar los componentes es preciso efectuar algunos cortes en la placa. Observe que sólo es necesario realizar cortes en un extremo de la placa o bien en el otro, según con qué ordenador desee utilizar la interface. Emplee un cuchillo puntiagudo o una pequeña sierra para metales para suprimir las porciones requeridas, tal como se indica. Todos los enlaces se efectúan en la cara superior de la placa utilizando alambre para enlaces de una sola hebra, con la excepción de tres enlaces entre patillas IC adyacentes. Estos enlaces se han de realizar haciendo correr soldadura entre los pares de patillas apropiados por el lado de cobre de la placa.

Comience por montar en la placa los componentes pasivos: las resistencias, los condensadores, los conectores DIN y los conectores DIL, tal como se indica. Efectúe los enlaces necesarios con el alambre de enlaces y monte el cristal de 2 MHz. El diodo debe orientarse de modo que el extremo marcado con la franja de color quede a la derecha (mirándolo desde arriba). Asegúrese de no fallar el pequeño enlace de debajo del condensador C3. Por último, coloque cuidadosamente los chips en su sitio, en sus respectivos conectores, prestando atención a la orientación de las muescas.

En el próximo capítulo del proyecto explicaremos de forma detallada el trabajo que aún resta por hacer antes de que se pueda conectar la interface a un ordenador y comprobar su funcionamiento

Diagrama de la disposición de los componentes







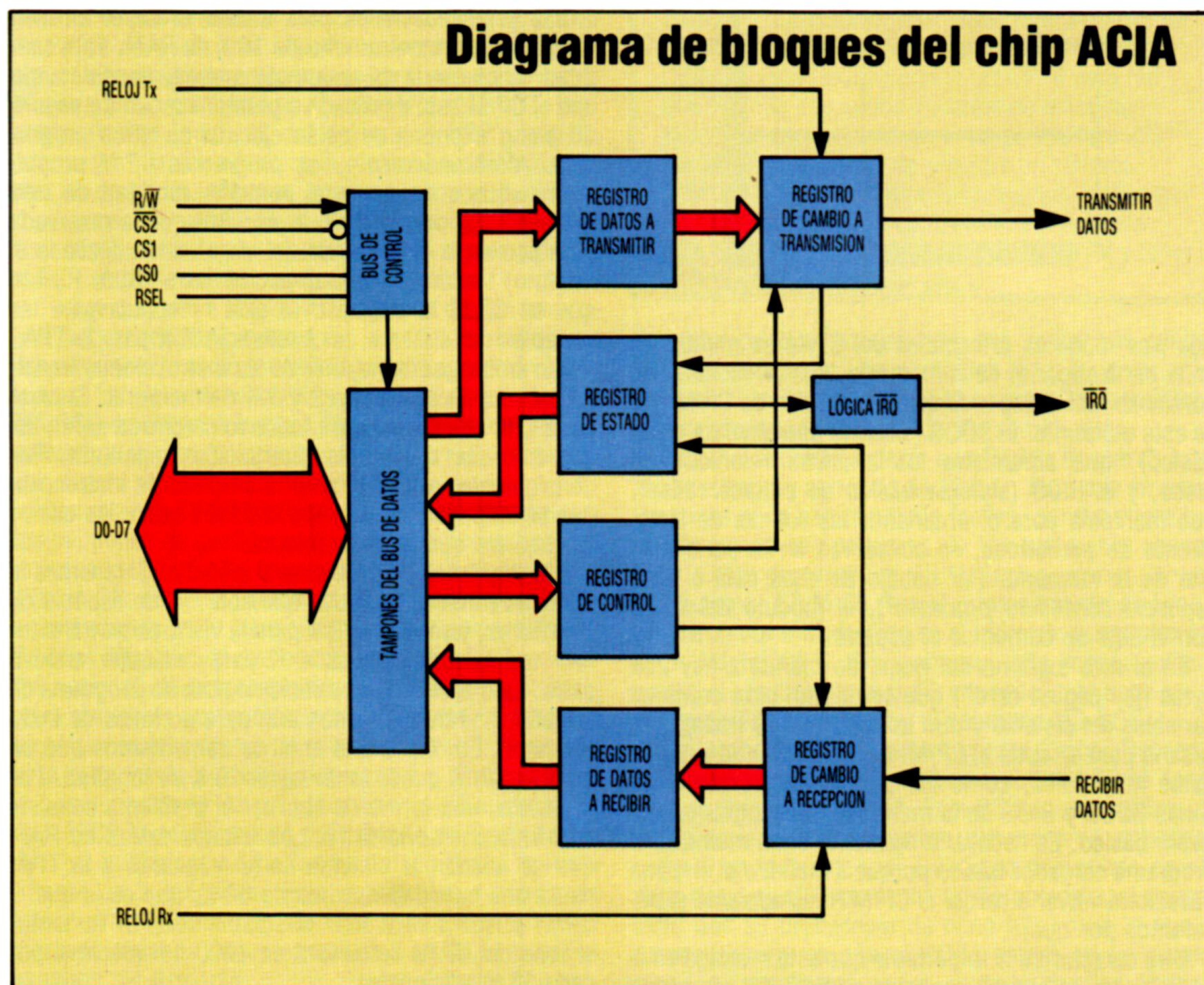
Otros componentes que necesitaremos en esta etapa son un reloj de interface, un optoaislante y conectores de entrada y salida, que constituyen la base de la interface. En el próximo capítulo reseñaremos de forma detallada las conexiones a la puerta del ordenador que se requieren para enlazar la interface con el Commodore 64 o el BBC Micro.

La estructura interna del dispositivo ilustrado se compone de un cierto número de registros de ocho bits. La línea de entrada R*egister* SEL*ect* (RSEL) permite el acceso a los registros desde el bus de datos. Aunque no es esencial comprender las funciones de los registros del ACIA para ejecutar el software que daremos, los que deseen programar la interface necesitarán conocer la siguiente información:

- El *registro de estado* se compone de ocho bits que describen el estado actual del chip ACIA. El programador los tiene a su disposición llevando a cabo una operación de lectura en el ACIA con la línea RSEL a nivel *low* (cero lógico).
- El *registro de control* contiene ocho bits que controlan la operación del ACIA. Al registro de control se accede escribiendo en el ACIA con la línea RSEL en un nivel *low*.
- El *registro de cambio a transmisión* (TSR: *transmit shift register*) realiza la conversión paralelo a serie que se requiere para transmitir un byte de datos. El registro se carga con un byte desde el *registro de datos a transmitir* (TDR: *transmit data register*) cada vez que el TDR está lleno y se haya completado la transmisión del byte anterior. Esta operación establece el bit 1 del registro de estado. Luego el byte es transportado a una velocidad determinada por el reloj de transmisión. Entonces a los ocho bits de datos se les añaden los bits de comienzo y final. Éste es un registro interno al cual no se puede acceder directamente desde el bus de datos.
- El *registro de datos a transmitir* (TDR) forma un tampón (o *buffer*) entre el TSR y el bus de datos del sistema. El byte a transmitir se carga en este registro escribiendo en el ACIA con la línea de selección de registros *high*. Esto limpia el bit 1 del registro de estado. Para no perder los datos, no se debe cargar el TDR cuando este bit está borrado. Ello se debe a que el byte previo está aún en el TDR esperando la transmisión y se escribiría sobre él.
- El *registro de cambio a recepción* (RSR: *receive shift register*) realiza la conversión de serie a paralelo que se requiere cuando se reciben datos desde la MIDI. Cuando el registro recibe un byte completo, carga los datos en el RDR, poniendo a 1 el bit de estado 0. Si ya estaba establecido a 1 con anterioridad, entonces también se establecerá a 1 el bit de estado 5, indicando que el byte anterior no había sido leído por la CPU y que los datos se perderán.

Si el byte recibido no tuvo la cantidad requerida de bits de comienzo y de final, entonces se establecerá a 1 el bit de estado 4. Esto podría ocurrir si en la línea de entrada en serie hubiera presente algún "ruido" eléctrico. Al igual que el TSR, al RSR no se puede acceder directamente.

- El *registro de datos a recibir* (RDR: *receive data register*) se carga cada vez que se recibe un byte completo desde el RSR. Contiene, en consecuencia, el byte de datos más reciente. A él se accede efectuando una operación de lectura del ACIA con la línea RSEL establecida *high*. Esta operación pone a 0 el bit de estado 0.

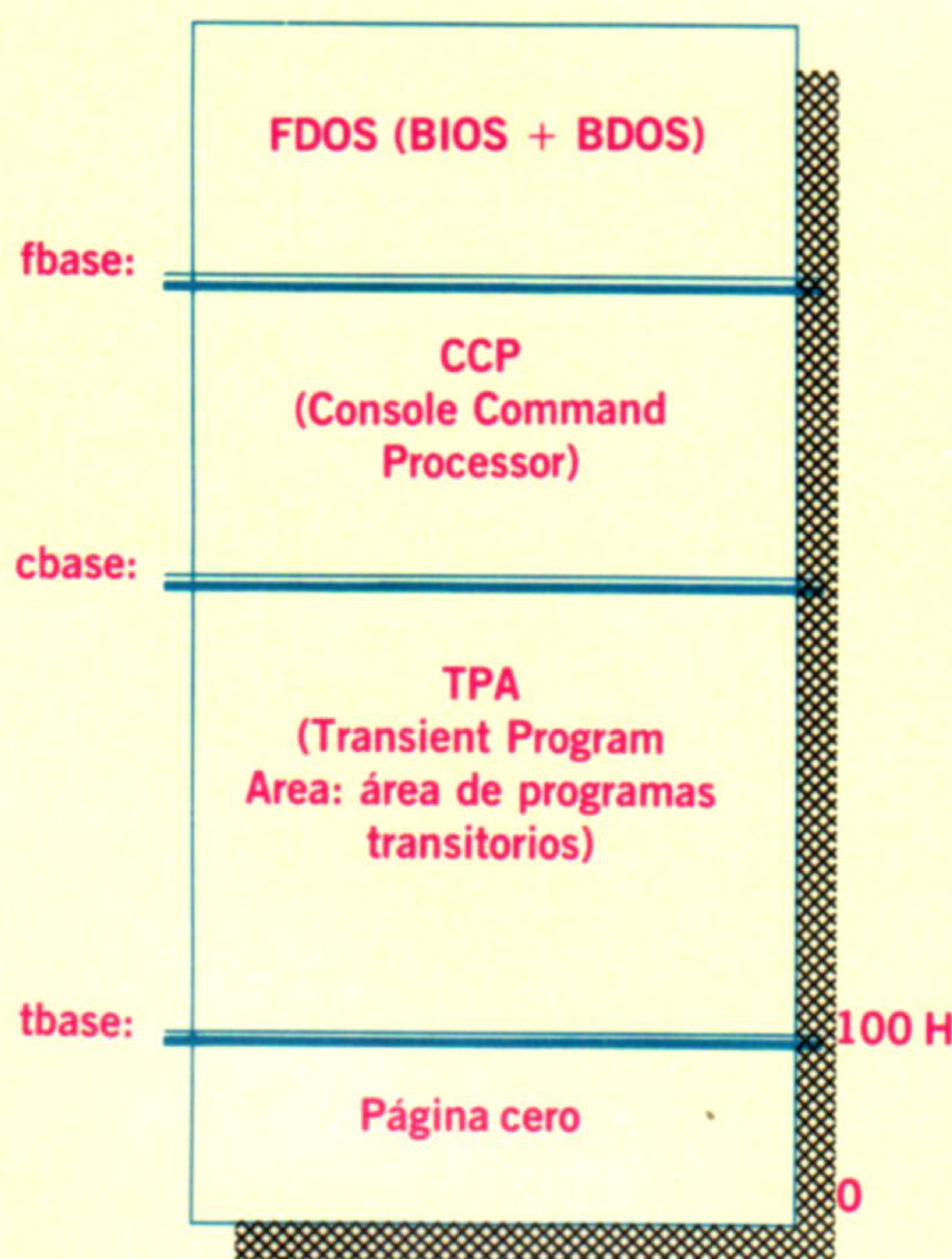


Futuro prometedor

En este último capítulo de nuestra serie acerca del CP/M, analizaremos cómo se organiza el sistema en la memoria del ordenador

Un espacio bien aprovechado

Este diagrama ilustra en qué lugar de la memoria se almacenan los diversos componentes del CP/M. En la parte inferior de la memoria, en la página cero, están las variables del sistema y los puntos de entrada, así como el programa cargador básico. Más arriba se halla la TPA (área de programas transitorios). La frontera superior de la TPA se puede alterar para permitir que los archivos que normalmente no cabrían en el área se sobreescriban en el CCP (*console command processor*). En la parte superior de la memoria están las rutinas BIOS y BDOS, que no se pueden sobreescribir porque son importantes para la ejecución de instrucciones



Los tres módulos principales del CP/M se mantienen en la parte superior de la memoria. Sus direcciones de comienzo reales dependerán de la versión de CP/M que se esté utilizando. El BDOS (sistema operativo de disco básico), que administra los archivos retenidos en disco, y el BIOS (sistema básico de entrada/salida), que manipula para el ordenador las rutinas de tratamiento de periféricos, se conservan en la parte más alta de la memoria. Por debajo de ellos está el CCP (*console command processor*), el módulo del CP/M con el cual se comunica el usuario.

En el otro extremo del mapa de memoria hay 256 bytes (la "página cero") que se utilizan para contener variables del sistema y otra información de trabajo necesaria para ejecutar el CP/M. Ésta comprende algunos datos muy útiles, como los puntos de entrada a las zonas BDOS y BIOS de la memoria y el programa cargador básico. Es necesario mantener en la memoria el programa cargador básico porque a menudo el sistema necesitará volver a cargar el CP/M (más adelante explicaremos por qué).

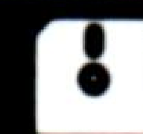
Otra característica importante que está retenida en la página cero es el área conocida como TFCB (*transient file control buffer:* buffer de control de archivos transitorios). En el capítulo anterior descubrimos que cuando el usuario pide que se llame un archivo desde disco, el CCP prepara en la memoria un bloque ficticio de control de archivo. Cuando el BDOS localiza un archivo, lo compara con el archivo retenido por el CCP y después pasa más información desde la pista del directorio al CCP. Esta información está almacenada en el TFCB.

Entre las secciones de la página cero y el CCP hay un área de memoria denominada TPA (*transient program area*: área de programas transitorios). Se trata de la zona que es como el espacio de trabajo del CP/M. Como hemos visto, cuando se somete una instrucción al sistema operativo, el CCP primero busca una pareja en la lista de las instrucciones residentes retenidas en el área CCP. Si no está entre ellas, el CCP da por sentado que la instrucción es transitoria y le ordena al BDOS que la localice. En el supuesto de que la instrucción esté residente en el disco de sistema, el BDOS cargará una copia del programa en código máquina en el principio del área de programas transitorios (TPA), que siempre está en la posición hexa 100 (el comienzo de la página uno en el mapa de memoria del ordenador). Una vez cargada, la instrucción está lista para ser ejecutada, lo que se realizará de forma automática.

Las especificaciones para utilizar el CP/M afirman que se requiere un mínimo de 16 K de RAM. Esta cantidad de memoria es realmente grande, considerando que el CP/M está destinado a permanecer en un segundo plano mientras se estén ejecutando otros programas. Afortunadamente, los programas CP/M propiamente dichos ocupan una pequeña cantidad de esta memoria. La mayor parte de la RAM queda reservada para uso de la TPA, la cual (si usted está utilizando el mínimo) terminará en la dirección hexa 2900. Puesto que en CP/M la página cero está reservada para las variables del sistema, se emplearán 7 K para la TPA.

Sin embargo, la mayoría de las instrucciones transitorias ocupan sólo entre 2 y 3 K de memoria. El resto de la TPA se utilizará para todos los archivos sobre los que necesite trabajar la instrucción transitoria. Por consiguiente, a la TPA sólo la emplean la instrucción que se esté ejecutando y los archivos sobre los cuales se requiere que actúe la misma.

Ello le plantea un problema al sistema. Si tenemos la cantidad mínima de RAM, que son 7 K de espacio de TPA libres, y se utilizan 3 K para la instrucción transitoria, nos quedan apenas 4 K para cualquier archivo extra. Esto apenas si es suficiente para un programa de tamaño moderado, menos aún para archivos de texto extensos. En el capítulo anterior comentamos que un archivo CP/M puede tener hasta 16 K de longitud. Por supuesto, una forma de abordar el problema consiste en añadir memoria extra. Los bancos extras de RAM que se añadan al sistema se le asignarán a la TPA, hasta una cantidad máxima de 64 K, que es la que el CP/M puede direccionar directamente. Por lo tanto, el tope del CP/M se elevará en 4000 hexadecimal por cada 16 K adicionales.



Pero supongamos que no podemos añadir la memoria extra requerida. ¿Cómo consigue el CP/M llevar a cabo esta tarea con menos de la memoria necesaria para ello? La respuesta es, simplemente, que todo sobreflujo proveniente de la TPA se sobreescribe (o *superpone*, en la jerga del CP/M), sobre el CCP. Esto no es tan drástico como parece. Por un lado, cuando se está ejecutando una instrucción no se requiere el CCP. El CP/M, al igual que todos los sistemas operativos, no aceptará ninguna otra instrucción mientras esté ejecutando la anterior. Por consiguiente, no es preciso que esté presente el CCP para interrumpirlas ni para ejecutar instrucciones residentes innecesarias.

Por supuesto, una vez terminado el programa de instrucción, será necesario volver a cargar los programas CCP con el objeto de preparar al sistema para aceptar la instrucción siguiente. Esto se efectúa como el último acto de una instrucción transitoria, y el programa llamará luego a la rutina de *carga de posiciones*, que se encuentra en la posición hexadecimal 0005 de la página cero. Esta dirección es el punto de entrada al sistema operativo que vuelve a cargar en la memoria el módulo CCP, listo para recibir la siguiente instrucción.

Cuando se inventó el CP/M, el hardware disponible (los chips de RAM, en especial) era muy costoso y, por consiguiente, eran pocas las máquinas equipadas con más de 16 K como estándar. De ese modo, todo el software diseñado para ejecutarse en esas máquinas había de hacerse a la medida para adecuarlo a las restricciones del hardware.

Con el fin de lograr que el CP/M fuera un OS lo más amplio posible para el espacio de memoria disponible, hubieron de superponerse algunos componentes. Dado que se requería que tanto el BIOS como el BDOS estuvieran residentes de forma permanente, los mismos no se podían utilizar. Muchas de las instrucciones se valían de ellos para acceder a archivos o efectuar alguna operación de entrada o salida (tales como escribir un archivo por una impresora o visualizarlo en una pantalla). Por tanto, se decidió que el CCP y la TPA "ocuparan" la misma zona de memoria, dado que el uno no se podía utilizar al mismo tiempo que el otro.

La zona CCP podía ser superpuesta por un programa transitorio; una vez ejecutado, éste, se volvía prescindible y el CCP se podía volver a cargar con seguridad.

Parece que ahora que las máquinas de gestión han comenzado a adoptar procesadores de 16 bits, el sistema operativo esté llamado a una inevitable desaparición. Pero aunque los sistemas operativos de 16 bits tales como el MS-DOS dominan en la actualidad el mercado de gestión, parece que el micro de ocho bits basado en el Z80 tiene aún por delante muchos años de utilidad.

Con frecuencia los sistemas de micros personales y los pequeños de gestión no necesitan la potencia de un procesador de 16 bits, pero sus unidades de disco exigen un OS. El CP/M, que durante tanto tiempo ha sido el estándar de facto para sistemas operativos de disco de ocho bits, parece una opción ideal para los fabricantes ansiosos de proporcionar una amplia base de software sin elevar sustancialmente los costos de desarrollo. Algunos fabricantes de ordenadores ya han elegido este camino, en particular Memotech y Amstrad.

Los usuarios del Amstrad tienen una dificultad especial. El CP/M se desarrolló para discos de 8 pulgadas y de 5 1/4 pulgadas, y Amstrad ha optado por el formato Hitachi de 3 pulgadas. Esto significa que habría de transcurrir algún tiempo antes de que aparezcan en el mercado muchos paquetes basados en CP/M para los usuarios del Amstrad.

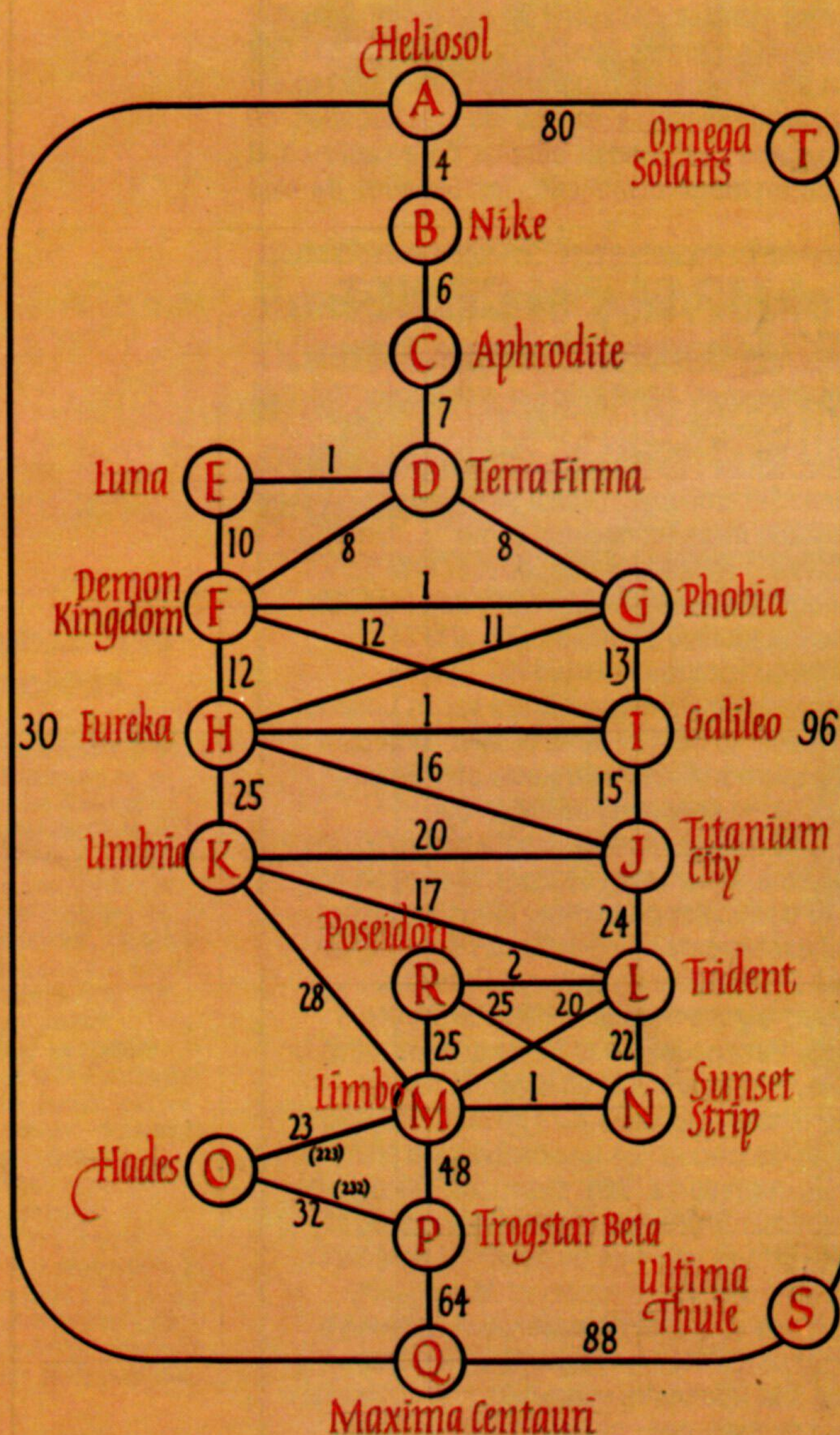
Otro problema del ordenador Amstrad es que el apoyo de su pantalla deja sólo 38 K de memoria libre para programas, que es muchísimo menos de lo que requieren algunos de los programas CP/M más recientes. Por consiguiente, a menos de que algunos de estos programas se puedan adaptar para caber en el espacio de memoria disponible, los usuarios de esta máquina pueden encontrarse con tener que utilizar software anticuado. No obstante, al "ir bajando en el mercado", parece que el CP/M de ocho bits continuará generando una gran base de consumidores.

Esto no equivale a decir que Digital Research haya abandonado su mercado de 16 bits. Pero a la empresa las cosas no le han resultado fáciles, porque ahora hay muchos más competidores de los que había cuando Gary Kildall desarrolló el CP/M. Una posterior versión del sistema, CP/M-86, fue el primer intento de Digital Research por introducirse en el mercado basado en los chips Intel 8088/6. Sin embargo, esta versión experimentó el rechazo general del mercado, que optó por la compatibilidad con IBM y el estándar MS-DOS.

Sin embargo, más recientemente Digital Research ha lanzado una versión multiusuarios del CP/M denominada Concurrent CP/M. Este OS lo puede utilizar un único usuario o bien varios ordenadores enlazados entre sí en forma de red. Quizá sea demasiado pronto para decir si el Concurrent CP/M obtendrá el mismo éxito que su predecesor de ocho bits, pero lo que sí parece seguro es que el CP/M estará en un primer plano, en una forma u otra, durante muchos años.

En el MP/M y el CP/NET

Aquí nos hemos dedicado exclusivamente a la versión 2.2 del CP/M, la versión más común del sistema operativo. Esta versión está diseñada para aplicaciones de un único usuario y una única máquina. Sin embargo, Digital Research ha desarrollado otros (conocidos como "*sistemas multitareas*") basados alrededor del CP/M para utilizar con numerosas máquinas y usuarios. Un sistema multitareas es uno en el cual la CPU comparte su tiempo (en virtud del *fraccionamiento del tiempo*) entre varios usuarios o periféricos. El OS que se ha desarrollado para permitir que numerosas personas utilicen la misma CPU se denomina MP/M (*multi-processing monitor control program*). Utilizando este sistema, un único ordenador se conecta con hasta 16 terminales diferentes, todos los cuales son completamente independientes entre sí. Una persona situada ante un terminal no tendrá conciencia alguna de la existencia de los otros usuarios. Para poder hacer frente a todos estos usuarios, se han incluido en el MP/M varias características diseñadas para ayudar a compartir ya sea los archivos o los dispositivos del sistema como las impresoras. Otro desarrollo del sistema CP/M que ha introducido Digital Research es el CP/NET. Ésta es una versión del CP/M para red que permite que varios usuarios de máquinas diferentes se comuniquen entre sí. A diferencia del MP/M, el CP/NET no requiere un sistema maestro con terminales subordinados, si bien hay un *nudo maestro* que controla la red. Además, este nudo maestro debe operar bajo MP/M y tener conectadas unidades de disco



Bernard Jennings

Contemplando las estrellas
El mapa interplanetario de los asistentes a las fiestas muestra las rutas hiperespaciales entre los planetas y sus respectivas duraciones. Se puede viajar entre planetas que no estén interconectados directamente por rutas hiperespaciales, pero los viajes realizados sin usar las rutas de la red consumen 1 000 unidades de tiempo cada uno. La tarea del programa *GENE* consiste en hallar las rutas más rápidas a través de la red, visitando cada uno de los veinte planetas

Código genético

Continuamos nuestra investigación acerca del aprendizaje de las máquinas

Cuando los científicos dedicados al estudio de la inteligencia artificial observaron la naturaleza en busca de ideas en las cuales basarse para diseñar sistemas que se mejoraran a sí mismos, tres cosas captaron su atención: el sistema nervioso, el sistema inmunológico y el proceso evolutivo. El sistema nervioso, en especial el cerebro humano, es un mecanismo de aprendizaje maravillosamente eficaz. Pero sus operaciones todavía son misteriosas: hemos de comprenderlo lo suficientemente bien para poder copiarlo. Se puede decir que también el sistema de respuesta inmunológica del cuerpo aprende, en tanto y en cuanto llega a distinguirse a sí mismo de otros, de modo que puede atacar y destruir cuerpos extraños. En el transcurso de una vida aprende a reconocer millones de millones de proteínas diferentes y, sin su sorprendente adaptabilidad y fiabilidad, moriríamos rápidamente. Y se trata de un sistema de memoria por repetición mecánica: sus poderes de generalización son rudimentarios.

El sistema evolutivo es, ciertamente, eficaz como medio de crear organismos cada vez más avanzados: puede que sea un poco lento para nuestros fines, pero puede ser acelerado en la simulación por ordenador. Por sobre todo, se lo comprende relativamente bien y es suficientemente simple como para que lo copiemos con ciertas esperanzas de éxito.

En el capítulo anterior explicamos que un enfoque "darwiniano" al aprendizaje de la máquina tenía algunas ventajas teóricas. Ahora veremos lo que sucede cuando lo ponemos en práctica.

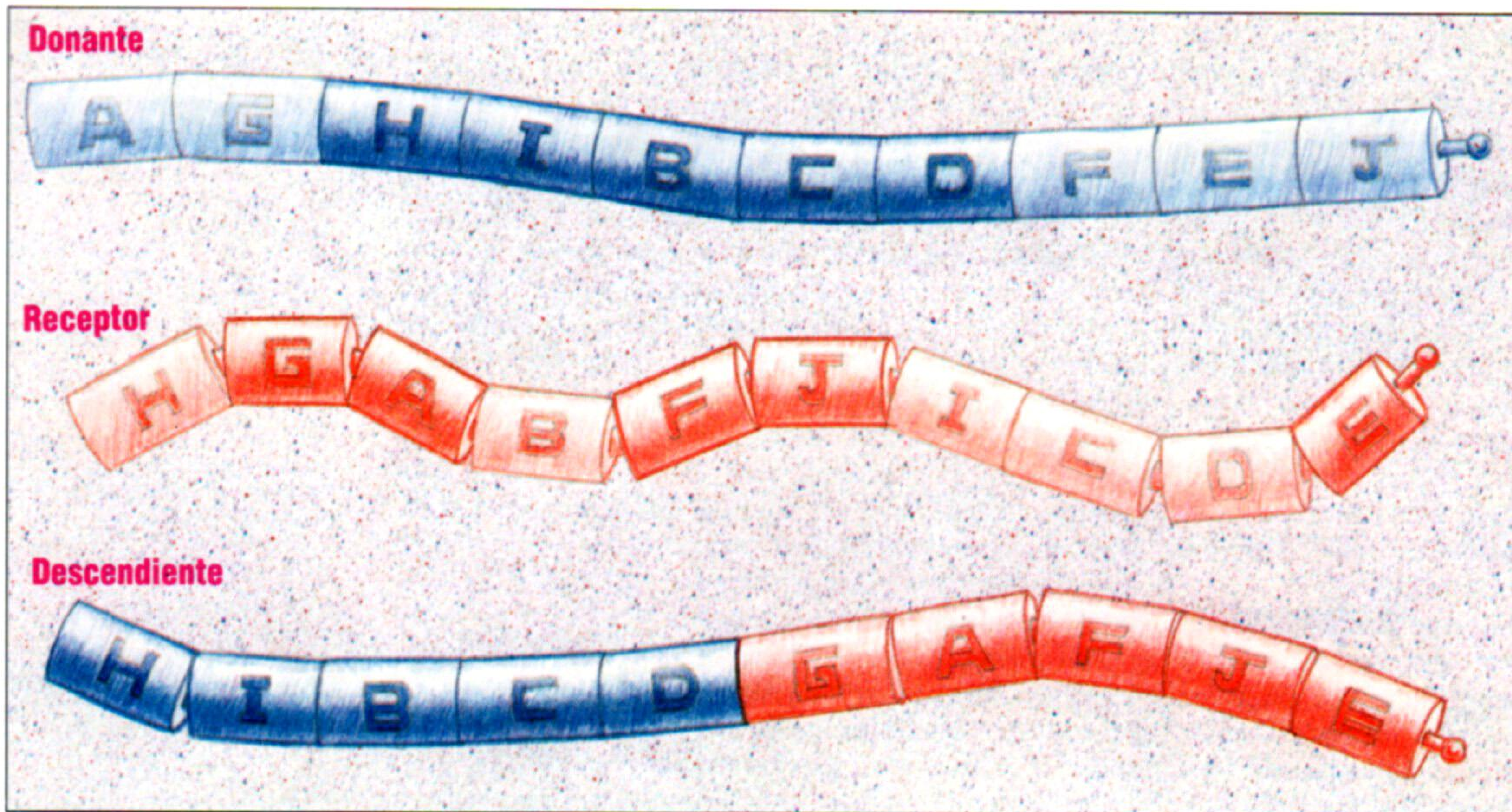
Para ilustrar el aprendizaje de la máquina utilizando un algoritmo evolutivo, analizaremos una versión del "problema del viajante de comercio" (TSP: *travelling salesman problem*). Este problema suele plantearse en el contexto de las 48 capitales de estado del sector continental de Estados Unidos (excluyendo Alaska y Hawai). El vendedor posee una tabla de las distancias entre las ciudades y ha de visitar cada una de las ciudades una sola vez y regresar a su punto de partida. El objetivo es minimizar la distancia total recorrida.

Parece engañosamente simple, ¡pero la cantidad de posibles recorridos o rutas es (N-1)!, donde N es la cantidad de ciudades. En un recorrido de 48 ciudades, las primeras paradas posibles son 47, seguidas de 46 posibles segundas paradas, seguidas de 45 posibles terceras paradas, y así sucesivamente. En realidad, ¡hay más rutas potenciales que átomos en el universo! Aun con 20 ciudades a visitar, ¡la cantidad de recorridos potenciales es de más de ciento veinte millones de millones!

Existen varios enfoques al TSP. Se lo puede tratar como un problema de búsqueda, empleando procedimientos similares a los que describimos en el segundo capítulo de esta serie. También se lo puede manejar con métodos Monte Carlo de acierto o error aleatorio. Sin embargo, vamos a seguir el poco ortodoxo enfoque de considerarlo como un problema de aprendizaje. Queremos un método que explore la enorme cantidad de soluciones potenciales de una forma económica. No esperaremos que nos ofrezca la solución óptima, pero sí esperamos hallar una buena solución.

El programa que ofrecemos aquí, *GENE* (*general evolutionary network explorer*: explorador de redes general y evolutivo) es un sistema de aprendi-





Un buen vástago
GENE cría nuevas rutas de la red emparejando dos buenas rutas ya existentes. Dado que éstas están retenidas como series en BASIC de 20 caracteres, el proceso de producción de un descendiente es esencialmente una sencilla manipulación de series. En este ejemplo, se toma de uno de los padres (el donante) la subserie MIBCD y se empalma con el otro padre (el receptor), asegurando de que no se duplique ninguna letra. Este proceso asegura que, al igual que en los verdaderos cruces genéticos, las características de los padres pasen a sus descendientes

zaje adaptado específicamente para explorar redes y desarrollar reglas que evolucionen para producir rutas cada vez más económicas a través de la red. Antes de pasar a describir de forma detallada cómo funciona el programa, necesitamos desarrollar una red a explorar. Podemos tomar un buen ejemplo de *The gatecrasher's guide to the galaxy* (Guía de la galaxia para intrusos), que lista, entre otras cosas, todos los planetas existentes en un radio de 80 años luz en los que usted puede encontrar una fiesta animada el sábado por la noche. Con la ayuda de este mapa podemos transformar el TSP en el PPCD (*planetary party crawl dilemma*: dilema de la ronda por las juergas interplanetarias), en el cual el objetivo es hacer acto de presencia en las 20 fiestas y regresar, esa misma noche, a su punto de partida.

El mapa ilustra las principales rutas hiperespaciales de nuestra localidad galáctica, con los tiempos que lleva el viaje entre cada parada. A los nudos que no están conectados por supercarreteras hiperespeciales se les otorga de forma arbitraria un costo de recorrido de 1 000 unidades de tiempo.

Aprendizaje evolutivo

En un sistema de aprendizaje evolutivo clásico hay una población de estructuras que se tratan como "seudoorganismos". Cada una de estas estructuras (a las que nos referiremos como *reglas*) define una solución potencial al problema en cuestión. También se utilizan para propagar nuevas estructuras (la *descendencia*) en formas que imitan algunas de las características de la reproducción biológica, tales como la transmisión de algunas de las características de los padres a sus hijos.

Dependiendo de su rendimiento en la tarea, se seleccionan reglas que es probable que sobrevivan durante el mayor tiempo y que tengan las mayores probabilidades de reproducirse. En *GENE*, los organismos/reglas tienen una estructura muy simple. Se retienen como series en el programa en BASIC. Por ejemplo, está:

ABJHMNCDKTSFRQEGILOP

en donde cada letra representa un planeta de la red planetaria y se produce una sola vez en toda la

Ingeniería alfabética

Primero se eligen al azar dos estructuras padre tomadas de entre las que han sobrevivido al proceso de selección (lo que implica, por tanto, que los padres deben ser más aptos para la tarea que la media). Una de estas estructuras se denomina *donante* (R1 %) y la otra "receptor" (R2 %) (ver líneas 3030-3060).

Se toma al azar un trozo de "material genético" (en realidad, una subserie) del donante y se coloca en la serie S$. Luego se llama a la subrutina 3300, que empalma el trozo del donante con el receptor, tomando todos los caracteres de la serie del receptor (excepto aquellos ya presentes en el donante) por el orden en el cual aparecen. El proceso de emparejamiento es, por tanto, asimétrico. Emparejar X con Y no produce el mismo resultado que emparejar Y con X, aunque las posiciones aleatorias P1 % y P2 % sean idénticas.

Veamos someramente un ejemplo a escala reducida. Dados los dos padres:

Donante: A G H I B C D F E J
Receptor: H G A B F J I C D E

podemos seleccionar la subserie

H I B C D

como la contribución del donante, y empalmar las letras restantes del receptor para producir:

H I B C D G A F J E

como resultado. Cabe consignar que ésta no es la única forma posible de cruzar un par de reglas-series. Quizá usted pueda pensar en otras. Pero asegura (como en los verdaderos cruces genéticos) que trozos de información de los padres pasen a la siguiente generación. También asegura que cada emparejamiento produzca un descendiente "válido". Esto significa que la rutina de limpieza de la línea 4000 del programa *GENE* no tiene razón de ser, pero la incluimos aquí para ofrecer un esbozo completo de éste.

serie. Por tanto, cada serie de 20 carácteres es una permutación de los planetas a visitar y define una ruta determinada alrededor de la red.

Es simple evaluar cada serie sumando la distancia necesaria para visitar los planetas por el orden especificado. Cuanto menor sea la distancia combinada para completar el recorrido, mejor será la ruta.

Disponemos que las series que sean peores que la media se supriman al cabo de cada "generación". Entonces se plantea el problema de cómo reemplazarlas. Obviamente, si seguimos la analogía biológica, llevaríamos a cabo algo similar a la reproducción sexual para sustituir las series descartadas. Pero la reproducción sexual no es el único medio de generar series nuevas. Alrededor del 8 % de las series supervivientes sufren mutaciones.

La subrutina de mutación, que empieza en la línea 3500, se limita a realizar una cierta cantidad de cambios al azar. Sin embargo, la mutación no es el operador genético primario, sino un operador de fondo que asegura que el sistema no se quede bloqueado en un nivel óptimo local. Las mejoras que se pueden conseguir en una generación sucesiva mediante métodos reproductivos puramente sexuales podrían tener un límite.

Si usted experimenta por sí mismo con el programa, descubrirá que el estándar medio de la población de reglas sí mejora con el tiempo, aunque esta progresión no es continua, porque hasta las mejores reglas a la larga pueden "morir". El programa en realidad hace una trampa, al preservar la mejor regla y sustituirla solamente por una aún mejor.

Usted puede "afinar" el sistema jugando con el coeficiente de muerte representado en la línea 2520. En esta línea, el elemento aleatorio establece el coeficiente de supervivencia de las reglas buenas de una generación a la siguiente en el 88 %, pero éste se puede alterar. Asimismo, puede regular el coeficiente de mutación en la línea 3520, así como introducir operaciones de mutación alternativos, tales como la inversión completa de una regla-serie. Subsisten dos preguntas:

1. ¿Cuán bueno es el método?
2. ¿Por qué funciona?

La primera pregunta se puede responder a través de la comparación con un enfoque Monte Carlo puro. Todos los algoritmos genéticos se modifican de hecho por procedimientos Monte Carlo. En un método Monte Carlo puro, se generan soluciones aleatorias y se conserva la mejor solución, todo ello dentro de un límite de tiempo específico. En este ejemplo, ello supondría probar una regla-serie aleatoria, evaluarla, conservarla si es la mejor disponible hasta ahora, y someterla a una mutación. El proceso repitiría todas las veces que se desea.

Si usted describe un programa como éste, descubrirá que rápidamente halla una solución bastante buena y que, a medida que transcurre el tiempo, se va mejorando a un nivel menor. Es probable que, al cabo de 20 000 intentos, la mejor solución sea sólo marginalmente mejor, en el mejor de los casos, que la solución tras 10 000 intentos. De forma global, los algoritmos genéticos mejoran para duraciones mayores y si consideramos el por qué de esto nos aproximaremos aún más a la respuesta para la segunda pregunta.

Un método Monte Carlo puro es esencialmente una búsqueda ciega. Un algoritmo genético, por el contrario, se vale de lo que encuentra para continuar la búsqueda. Los patrones específicos que contribuyen a un buen rendimiento se preservan y se propagan a través de la base de conocimiento (la población de reglas) y se vuelven a combinar en contextos ligeramente diferentes. De hecho, la búsqueda se dirige preferentemente hacia regiones del "espacio (multidimensional) del problema" en donde se hayan hallado buenos resultados. A menos que la función de evaluación sea extraordinariamente discontinua, cabe esperar que esto conduzca a una razonable estrategia de búsqueda.

El programa GENE

Las principales estructuras de datos que se utilizan para implementar nuestro sistema de aprendizaje evolutivo corresponden a las siguientes matrices:

NOMBRE$(TAMAÑO%)	Nombres de los nudos de la red
ENLACE%(TAMAÑO%, TAMAÑO%)	Distancia entre los nudos
NX%(TAMAÑO%)	Utilizada en la rutina de emparejamiento
R$(NR%)	Las reglas propiamente dichas (series)
RV(NR%)	Los valores de cada regla

NX% se emplea cuando se empalma la contribución del receptor a la nueva regla que se está creando (rutina 3300), de modo tal que cada nudo/letras aparezca una vez en el descendiente.

R$ retiene la actual población de reglas como series de longitud TAMAÑO%. Cada una define una recorrido particular.

RV da el "valor" de cada regla correspondiente a la distancia del recorrido que representa. Si RV(R%)=0, la regla R% estará "muerta" y será necesario sustituirla en la siguiente generación. (No son posibles recorridos de costo cero.)

TAMAÑO% se puede alterar si usted desea probar mapas de redes diferentes, en cuyo caso necesitará alterar desde la línea 8000 en adelante. NR% se puede ajustar para ver el efecto de una o más reglas sobre la población

```
10 REM ******************
11 REM **       GENE      **
13 REM ******************
100 GOSUB 1000 : REM preparar matriz mapa
101 G%=0: SN%=0
105 B=TAMAÑO%*1000: B$=""
110 INPUT "Cuantas generaciones?", MG%
111 GOSUB 1700 : REM reglas iniciales.
115 IF SN%=0 THEN INPUT "El nudo de partida es el No. ",SN%
120 REM **** BUCLE PRINCIPAL DEL PROGRAMA ****
130 G%=G%+1
140 PRINT "Generación ";G%
150 GOSUB 2000 : REM evaluar reglas
160 GOSUB 2500 : REM matar reglas malas
170 GOSUB 3000 : REM emparejar reglas buenas
180 GOSUB 3500 : REM mutaciones
190 GOSUB 4000 : REM limpieza
200 IF G%<MG% THEN 120
220 GOSUB 5000 : REM volcar reglas nuevas
250 END
999 :
1000 REM — RUTINA PARA PREPARAR MAPA-RED:
1001 TAMAÑO%=20: NR%=24
1010 DIM NOMBRE$(20), ENLACE%(20,20)
1011 DIM NX%(TAMAÑO%)
1012 DIM R$(NR%),RV(NR%)
1013 REM reglas y valores de las reglas
1015 FOR I%=1 TO TAMAÑO%
1020 FOR J%=1 TO TAMAÑO%
1022 ENLACE%(I%,J%)=1000 : REM defecto
1023 IF I%=J% THEN ENLACE%(I%,J%)=0
1025 NEXT: NEXT
1030 NC%=0: L%=0
1032 FOR I%=1 TO NR%: RV(I%)=0: NEXT
1033 RESTORE
1040 REM **** LEER NOMBRE Y NUM DE NUDO *
1050 READ N$, ID%
1055 PRINT N$, ID%
1060 IF ID%< >0 THEN GOSUB 1500
1070 IF ID%<>0 THEN 1040
1080 PRINT NC%;" LECTURA POSICIONES IN."
1088 PRINT L%;" enlaces no por defecto."
1090 RETURN
1100
1500 REM — NUDO INDIVIDUAL Y CONEXIONES :
1510 NC%=NC%+1
1520 IF ID%<> NC% THEN PRINT"ATENCION: Nu
1530 NOMBRE$(NC%)=N$
1550 REM **** PREPARAR MATRIZ DISTANCIAS
1560 READ NI%,NT%
1570 ENLACE%(ID%,NI%)=NT%
1588 REM los enlaces cero no importan.
1590 L%=L%+1
1600 IF NI%>0 THEN 1550
1610 RETURN
1620 :
1700 REM — Reglas ficticias iniciales:
1710 S$=LEFT$("ABCDEFGHIJKLMNOPQRSTUVW
1715 GOSUB 1780 : REM leer archivo si hay
1720 FOR R%= 1 TO NR%
1730 R$(R%)=S$
1740 PRINT S$;R%
1750 I%=INT(RNS(1)*TAMAÑO%+1): J%=INT(R
1760 GOSUB 6000 : REM TRUEQUE
1770 NEXT
1775 RETURN
1777 :
1780 INPUT "Archivo de reglas viejo (RETURN si n
1790 IF RF$=""THEN RETURN
1800 F%=OPENUP(RF$)
1810 INPUT # F%, B$, B, SN%
1820 CLOSE # F%
1825 S$=B$
1830 REM solo el de arriba.
1840 RETURN
1850 :
2000 REM — Rutina de evaluacion de reglas:
2010 T=0
2020 FOR R%=1 TO NR%
2030 IF RV(R%) <=0 THEN GOSUB 2200
2040 T=T+RV(R%)
2050 NEXT
2060 AV=T/NR% : REM valor medio
2070 PRINT"Marcador medio = ";AV
2080 RETURN
2100 :
2200 REM — Evaluacion de una sola regla:
2210 S$=R$(R%)
2220 P1%=SN% : REM nudo de partida
2230 GT%=0
2240 FOR S%=1 TO LEN(S$)
2250 P2%=ASC(MID$(S$,S%,1))-64
2260 IF P2%=SN% THEN GOTO 2290
2270 GT%=GT%+ENLACE%(P1%,P2%)
2280 P1%=P2%
2290 NEXT
2300 RV(R%)=GT%+ENLACE%(P2%,SN%)
2310 RETURN
2320 :
2500 REM — Rutina para matar reglas malas:
2510 FOR R%=1 TO NR%
2515 IF RV(R%) < B THEN B=RV(R%): B$=R$(R
2520 IF RV(R%) > AV OR INT(RND(1)*100+1)>
     RV(R%)=0
2530 NEXT
2540 RETURN
2550 REM — Son mejores los valores menores NB
2560 :
3000 REM — Rutina de emparejamiento
3010 FOR R%=1 TO NR%
3020 IF RV(R%)> 0 THEN GOTO 3120
3030 R1%=INT(RND(1)*NR%+1):IF RV(R1%)<
3050 R2%=INT(RND(1)*NR%+1):IF RV(R2%)<
3070 REM "padres"elegidos.
3075 P2%=INT(RND(1)*(TAMAÑO%-1)+1)
3080 P1%=INT(RND(1)*(TAMAÑO%+1) : REM p
3090 S$=MID$(R$(R1%),P1%,P2%)
3100 GOSUB 3300 : REM empalmar resto
3110 R$(R%)=S$
3120 NEXT
3140 RETURN
3150 :
3300 REM — Rutina empalme de genes!
3310 FOR S%=1 TO TAMAÑO%
3320 NX%(S%)=0: NEXT
3330 FOR S%=1 TO LEN(S$)
3340 SX%=ASC(MID$(S$,S%,1))-64
3350 NX%(SX%)=NX%(SX%)+1
3360 NEXT
3370 FOR S%=1 TO LEN(R$(R2%))
3380 SX%=ASC(MID$(R$(R2%),S%,1))-64
3390 IF NX%(SX%)>0 THEN GOTO 3420
3400 S$=S$+MID$(R$(R2%),S%,1)
```

```
3410 NX%(SX%)=NX%(SX%)+1
3420 NEXT
3440 RETURN
3450 :
3500 REM — Rutina de mutacion:
3510 FOR R%=1 TO NR%
3520 IF RND(100)>8 THEN GOTO 3580
3522 S$=R$(R%)
3525 FOR T%=1 TO 7
3530 R1%=INT(RND(1)*TAMAÑO%+1)
3540 R2%=INT(RND(1)*TAMAÑO%+1)
3560 I%=R1%:J%=R2%:GOSUB 6000:REM TRUEQUE
3565 NEXT T%
3570 R$(R%)=S$
3575 RV(R%)=0
3580 NEXT
3590 REM por ahora solo trocar.
3595 REM tambien necesita inversion.
3600 RETURN
3620 :
4000 REM — Rutina de limpieza:
4010 RETURN
4020 REM ficticia por ahora.
4040 :
5000 REM — Impresion de resultados:
5010 PRINT"Las rutas son:"
5020 BR=TAMAÑO%+1000
5030 R%=0
5040 FOR I%=1 TO NR%
5050 IF RV(I%)=0 THEN 5100
5060 PRINT I%,RV(I%)
5070 PRINT R$(I%)
5080 IF RV(I%)<BR THEN BR=RV(I%): R%=I%
5100 NEXT
5105 A$=GET$
5110 PRINT
5120 PRINT"La mejor es:"
5130 PRINT R$(R%),R%
5131 S$=R$(R%):GOSUB 6400
5133 PRINT"Distancia = ";RV(R%)
5134 A$=GET$
5135 PRINT"La mejor de todas :"
5136 PRINT B$
5140 S$=B$:GOSUB 6400
5143 PRINT"Distancia 8 ";B
5144 A$=GET$
5145 PRINT
5148 GOSUB 5500: REM vuelco archivo
5150 RETURN
5160 :
5500 REM — Rutina de vuelco:
5510 INPUT "Nuevo archivo de reglas (RETURN si (no hay) ninguno)",RF$
5520 IF RF$=" " THEN RETURN
5530 F%=OPENOUT(RF$)
5540 PRINT # F%,B$,B,SN%
5550 CLOSE # F%
5555 REM solo la mejor.
5560 RETURN
5570 :
6000 REM **** TROCAR DOS CARACTERES DE S$ ****
6040 IF I%>J% THEN T%=I%: I%=J%: J%=T%
6050 X$=MID$(S$,I%,1)
6060 Y$=MID$(S$,J%,1)
6070 S$=LEFT$(S$,I%-1)+Y$+MID$(S$,I%+1)
6080 S$=LEFT$(S$,J%-1)+X$+MID$(S$,J%+1)
6090 RETURN
6100 :
6400 REM **** VIAJE ****
6430 PRINT " 0 ";NOMBRE$(SN%)
6440 FOR I%=1 TO LEN(S$)
6450 N%=ASC(MID$(S$,I%,1))-64
6460 IF N%<>SN% THEN PRINT I%;" ";NOMBRE$(N%)
6470 NEXT
6480 PRINT I%;" ";NOMBRE$(SN%)
6490 RETURN
6500 :
8000 REM — DATOS PARA MAPA INTERPLANETARIO:
8010 DATA HELIOSOL, 1
8020 DATA 2,4, 17,30, 20,80, 0,0
8030 DATA NIKE,2
8040 DATA 1,4, 3,6, 0,0
8050 DATA APHRODITE,3
8060 DATA 2,6, 5,7, 0,0
8070 DATA LUNA,4
8080 DATA 5,1, 6,10, 0,0
8090 DATA TERRA FIRMA,5
8100 DATA 3,7, 7,8, 6,8, 4,1, 0,0
8110 DATA DEMON KINGDOM,6
8120 DATA 5,8, 7,1, 9,12, 8,12, 5,10, 0,0
8130 DATA PHOBIA, 7
8140 DATA 5,8, 9,13, 8,11, 6,1, 0,0
8150 DATA EUREKA, 8
8160 DATA 6,12, 7,11, 9,1, 10,16, 11,25, 0,0
8170 DATA GALILEO, 9
8180 DATA 10,15, 8,1, 6,12, 7,13, 0,0
8190 DATA TITANIUM CITY, 10
8200 DATA 9,15, 12,24, 11,20, 8,16, 0,0
8210 DATA UMBRIA, 11
8220 DATA 8,25, 10,20, 12,17, 13,28, 0,0
8230 DATA TRIDENT, 12
8240 DATA 10,24, 14,22, 13,20, 11,17, 18,2, 0,0
8250 DATA LIMBO, 13
8260 DATA 12,20, 14,1, 16,48, 15,23, 11,28, 0,0
8270 DATA SUNSET STRIP, 14
8280 DATA 12,22, 13,1, 18,25, 0,0
8290 DATA HADES, 15
8300 DATA 13,223, 16,232, 0,0
8310 DATA TROGSTAR BETA, 16
8320 DATA 13,48, 17,64, 15,32, 0,0
8330 DATA MAXIMA CENTAURI, 17
8340 DATA 16,64, 1,30, 19,88, 0,0
8350 DATA POSEIDON, 18
8360 DATA 12,2, 14,25, 13,25, 0,0
8370 DATA ULTIMA THULE, 19
8380 DATA 17,88, 20,96, 0,0
8390 DATA OMEGA SOLARIS, 20
8400 DATA 1,80, 19,95, 0,0
8410 DATA NINGUNLUGAR, 0
```

Complementos al BASIC

El programa *GENE* está escrito para el BBC Micro.

Commodore 64:

Introduzca las siguientes modificaciones:

```
1820 CLOSE 2
5105 GET A$:IF A$=" " THEN 5105
5134 GET A$:IF A$=" " THEN 5134
5144 GET A$:IF A$=" " THEN 5144
5550 CLOSE 2
```

Para sistemas de disco añada estas líneas:

```
1800 OPEN 2,8,2,RF$+",S,R":F%=2
5530 OPEN 2,8,2,RF$+",S,W":F%=2
```

o éstas para sistemas de cassette:

```
1800 OPEN 2,1,0,RF$:F%=2
5520 OPEN 2,1,1,RF$:F%=2
```

Spectrum:

Suprima el signo % en todas las variables, sustituya NOMBRE$() por N$(), ENLACE%(,) por L(,), NX%() por N(), RV() por R(), TAMAÑO% por SI y RF$ por F$ en todo el listado, y encierre entre comillas los nombres de los planetas de las líneas DATA. Introduzca las siguientes modificaciones:

```
1010 DIM N$(20,15),L(20,20)
1012 DIM R$(NR,20),R(NR),D$(3,25)

1710 LET S$="ABCDEFGHIJKLMNOPQRST
     UVWXYZ",(TO SI)

1800 LOAD F$ DATA D$()
1810 LET B$=D$(1)
1820 LET B=VAL(D$(2)):LET SN=VAL(D$,3))

2250 LET P2=CODE S$(S TO S)-64
3090 LET S$=R$(R1)(P1 TO P2-P1)
3340 LET SX=CODE S$(S TO S)-64
3400 LET S$=S$+R$(RS)(S TO S)

5105 IF INKEY$=" " THEN GO TO 5105
5134 IF INKEY$=" " THEN GO TO 5134
5144 IF INKEY$=" " THEN GO TO 5144

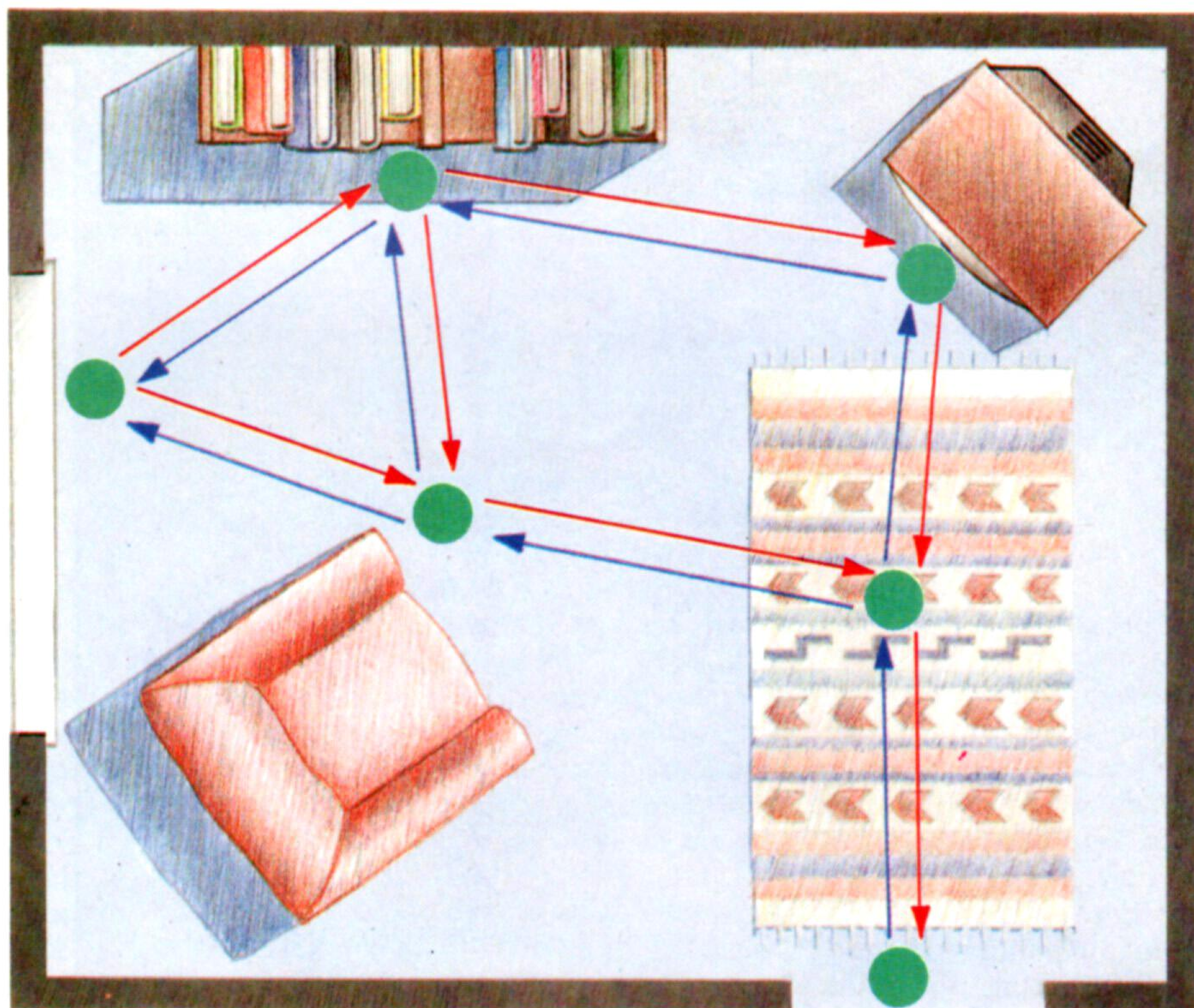
5530 LET D$(1)=B$:LET D$(2)=STR$(B)
5540 LET D$(3)=STR$(SN)
5550 SAVE F$ DATA D$()

6050 LET X$=S$(I TO I)
6060 LET Y$=S$(J TO J)
6070 LET S$=S$(TO I-1)+Y$+S$(I+1 TO)
6080 LET S$=S$(TO J-1)+X$+S$(J+1 TO)

6450 N=CODE S$(I TO I)-64
```

Prueba de capacidad

Espacio para moverse
El entorno del robot es un conjunto de posiciones "conocidas", que se han entrado como hechos en su base de datos. Éstas se enlazan mediante un conjunto de "caminos" a lo largo de los cuales se puede desplazar y que se almacenan en el formato "camino(tv,estantería)", y así sucesivamente.
En nuestro programa *Simulación de robot móvil* ilustramos cómo utiliza el robot este conocimiento para trazar un camino desde A hasta B



Concluiremos esta serie dedicada al PROLOG con una evaluación de la capacidad del lenguaje para utilizar sus propios programas como datos e ilustrando su idoneidad para la programación de inteligencia artificial

Los japoneses han elegido al PROLOG como "lenguaje central" para su proyecto de ordenadores de quinta generación. Ello se debe a dos motivos fundamentales: Los términos del PROLOG pueden tener una forma muy similar a las "relaciones" de una base de datos relacional, y para los japoneses la base de su sistema se considera una sofisticada máquina de base de datos.

En segundo lugar, el PROLOG es un lenguaje ideal para escribir programas de inteligencia artificial (AI). Ello se debe a su capacidad para usar sus programas como datos, y a que su intérprete se parece a los "motores de inferencia" de muchos sistemas modernos.

Para ver la utilidad que puede tener utilizar sentencias de programas como datos, ofrecemos un programa que es una sencilla simulación de un robot móvil que se desplaza por la habitación de una casa. Este programa ilustra las características más avanzadas del PROLOG y nos dará una noción del aspecto que tienen los programas en este lenguaje y cómo se comportan.

La habitación se define como un conjunto de lugares y un conjunto de caminos que los unen. Queremos que nuestro robot obedezca instrucciones, tales como "ir desde el televisor hasta el sillón", y que encuentre por sí mismo la ruta más corta. En primer lugar, consideremos el objetivo ir(Allí), que significa: "ir desde donde estés hasta Allí por la ruta más corta posible".

En el programa hay dos cláusulas para ir(Allí). La primera es para detectar el caso trivial planteado cuando el robot ya está Allí. Observe que para esto necesitamos un hecho de la base de datos (en(puerta)), que registra la posición actual del robot. La segunda cláusula utiliza este hecho para establecer una posición de partida, y después llama a otro procedimiento: ir(Lugar1,Lugar2).

Dado que ir(Lugar1,Lugar2) se puede usar separado de ir(Lugar), primero comprueba que Aquí y Allí sean lugares que conoce y que Aquí sea en realidad la posición actual. Si falla alguna de estas comprobaciones, el PROLOG abandona la cláusula actual y pasa a la siguiente. La segunda cláusula ir(Aquí,Allí) está para trasladar al robot a Aquí en el caso de que no se hallara ya en esa posición. Esto lo hace con una llamada a ir(Aquí) y, cuando ésta ha triunfado, con otra llamada a ir(Allí). La lectura procesal de esta cláusula es: "para ir desde (Aquí) hasta (Allí) si no estás ya (Aquí), primero ir (Aquí) y luego ir (Allí)". La tercera cláusula para ir(Aquí,Allí) se incluye sólo para imprimir un mensaje de error si no se conoce ninguno de los dos lugares.

El predicado "Findall"

Suponiendo que triunfen las comprobaciones preliminares para el primer ir(Aquí,Allí), el siguiente subobjetivo será findall. Éste es un predicado que en ocasiones ya está incorporado en el PROLOG pero que, de no ser así, se puede añadir (al igual que el FORTH, el PROLOG se puede ampliar fácilmente definiendo nuevos predicados, puesto que los predicados definidos por el usuario poseen el mismo status que los predicados del sistema). Findall posee tres argumentos. El primero es un nombre de variable, el segundo es un objetivo y el tercero es una lista de variables (RL).

Funciona en PROLOG intentando demostrar el objetivo que se le ha dado (plan(Aquí,Allí,[],Ruta), en este caso) todas las veces que pueda. La variable del primer argumento debe concordar con alguna de las del objetivo y, cada vez que éste triunfe, el valor que se había concretado para esa variable se añade a la lista del tercer argumento.

Por ejemplo, el objetivo plan(Lugar1,Lugar2,[],Ruta) hallará una ruta entre Lugar1 y Lugar2 y la colocará en la variable Ruta. De modo que, en este caso, findall reúne todas las rutas que se puedan hallar entre Aquí y Allí y las coloca en una lista.

Llegados a este punto, hemos de destacar que aún no hemos necesitado definir el procedimiento para findall. De hecho, es una práctica normal cuando se programa en PROLOG desarrollar programas de esta forma. Debido a su naturaleza declarativa, escribimos nuestros procedimientos "de arriba hacia abajo", definiendo primero los objetivos de alto nivel y rellenando los detalles.

El predicado shortest(RL,Ruta Corta) es otro predicado que todavía no hemos definido aquí. Su tarea consiste en tomar una lista de listas (Lista1) y crear una



nueva lista (Lista2) que contenga sólo la lista más corta que encuentre. Puesto que todas las listas de Lista1 son rutas, la lista más corta será la ruta más corta.

Hemos llegado a una etapa en la que nuestro robot ha planeado la ruta más corta para llegar a su destino; todo cuanto resta por hacer es que se desplace hasta allí. Esto se consigue mediante otro procedimiento "ficticio", caminar(Ruta). Hemos definido caminar de modo que simplemente escriba las rutas, pero la ruta en realidad es una lista de objetivos del PROLOG. Éstos se le podrían haber dado al PROLOG para que los ejecutara como un programa en lugar de imprimirlos en la pantalla. Se podría definir el predicado mirar(Lugar) de modo que efectuara una exploración por sensor para localizar Lugar y luego hiciera girar al robot, definiendo al mismo tiempo mover(Lugar1,Lugar2) de modo que permitiera que el robot se desplazara entre los dos puntos.

Este programa de desplazamiento lo escribe por sí mismo el programa principal dentro del procedimiento plan. Esto se hace simplemente. En esencia, el algoritmo que utiliza es: para hallar una ruta de A a B, primero hallar una ruta de A a C, después hallar una ruta de C a B. El procedimiento plan es recursivo. La primera cláusula está allí para detener la recursión cuando se alcance el punto de destino (es decir, siempre hay una ruta desde A hasta A).

La segunda cláusula es en realidad la que hace todo el trabajo. Su lectura declarativa es que existe un plan para ir desde A hasta B por la ruta R si:

1. hay un camino desde A hasta C, *y*
2. C no está incluido en la lista ya visitada, V (luego se añadirá C a esta lista para impedir que en el futuro se avance en círculo), *y*
3. hay un plan para ir desde C hasta B por la ruta R1.

Cuando todas estas condiciones son verdaderas, la ruta final, R, es la lista que contiene un programa para ir de A a C añadido a la ruta-hasta-ahora (retenida en R1). Los procedimientos recursivos como éste son muy difíciles de seguir (intente seguirlo mediante lápiz y papel), pero le confieren al PROLOG una extraordinaria concisión.

Para completarlo es preciso actualizar la posición actual del robot. Ello se realiza con los predicados incorporados retract y assert. retract(X) suprime de la base de datos la primera cláusula que concuerda con X, y asserta(X) añade a la base de datos la cláusula X como la primera de ese tipo (assertz(X) añade X como la última). Estos predicados son poderosas herramientas para la programación de AI.

Simulación de robot móvil
Versión en PROLOG estándar:

```
en(puerta).           /*la posición actual del robot*/

ir(Allí):-            en(Allí),write('Ya estoy allí!'),nl,nl,nl.

ir(Allí):-            en(Aquí),ir(Aquí,Allí).

ir(Aquí,Allí):-       lugar(Aquí),lugar(Allí),en(Aquí),findall(Ruta,plan(Aquí,Allí,[],Ruta),RL),shortest(RL,
                      RutaCorta),caminar(RutaCorta),retract(en(Aquí)),asserta(en(Allí)),decirdonde.

ir(Aquí,Allí):-       not(en(Aquí)),write('No estoy en el'),write(Aquí),write('de modo que iré hasta allí
                      primero.'),nl,nl,ir(Aquí),ir(Allí).

ir(Aquí,Allí)         write('Sólo puedo visitar los lugares que sé que existen'),nl,nl.

plan(A,A,_,R).
plan(A,B,V,R):-       camino(A,C),not(member(C,V)),append[C],V,V1),plan(C,B,V1,R1),append([mirar(C),
                      mover(A,C)],R1,R).

caminar(Ruta):-       write(Ruta),nl,nl.      /*a definir por completo luego*/

decirdónde:-          en(Lugar),write('Estoy en el'),write(Lugar),nl,nl,nl.

/*una lista de los caminos que conoce el robot*/    /*y una lista de los lugares que conoce*/

camino(estantería,sillón).                          lugar(puerta).
camino(sillón,estantería).                          lugar(alfombra).
camino(estantería,tv).                              lugar(tv).
camino(tv,estantería).                              lugar(estantería).
camino(tv,alfombra).                                lugar(ventana).
camino(alfombra,tv).                                lugar(sillón).
camino(alfombra,sillón).
camino(sillón,alfombra).
camino(puerta,alfombra).
camino(alfombra,puerta).
camino(ventana,sillón).
camino(sillón,ventana).
camino(ventana,estantería).
camino(estantería,ventana).
```

Versión en Micro-PROLOG:

```
(en puerta)           /*la posición actual del robot*/

((ir X)               (en X) (P Ya estoy allí!) PP,PP,PP)

((ir X)               (en Y) (ir Y X))

((ir X Y)             (lugar X) (lugar Y)(en X)
                      (findall Z (plan XY()Z)x)
                      (shortest x y)
                      (caminar y)
                      (DELCL ((enX))) (ADDCL ((en Y)))
                      (decirdónde))
((ir X Y)             (NOT en X)
                      (P No estoy en el) (PX)
                      (P de modo iré allí primero.) PP PP
                      (ir X) (ir Y))
((ir X Y)             (P Sólo puedo visitar lugar que sé que existen) PP PP)
(plan X X x1 Z)
((plan X Y x Z)       (camino X X1) (NOT member X1 x) (append (X1) xx1)
                      (plan X1 Y x1 Z1)
                      (append ((mirar X1)(mover X X1))Z1 Z))

((caminar Z)          (P Z) PP PP)
((decirdónde)         (en X) (P Estoy en el X) PP PP PP)

/*una lista de los caminos que conoce el robot*/    /*y una lista de los lugares que conoce*/

(camino estantería sillón)                          (lugar puerta)
(camino sillón estantería)                          (lugar alfombra)
(camino estantería tv)                              (lugar tv)
(camino tv estantería)                              (lugar estantería)
(camino tv alfombra)                                (lugar ventana)
(camino alfombra tv)                                (lugar sillón)
(camino alfombra sillón)
(camino sillón alfombra)
(camino puerta alfombra)
(camino alfombra puerta)
(camino ventana sillón)
(camino sillón ventana)
(camino ventana estantería)
(camino estantería ventana)
```

Maniobras programadas
La información acerca del entorno del robot la entra el usuario en la base de datos del PROLOG. El PROLOG utiliza el procedimiento ir(lugar1,lugar2) para construir una ruta desde A hasta B, lo que hace hallando primero una ruta desde A a C y luego otra desde C hasta B. Observe el empleo del símbolo de subrayado como un argumento dentro de la cláusula plan(A,A,_,B). El símbolo se puede emplear en PROLOG como una *variable anónima* o *máscara* para la cual no se concretará ningún valor cuando se ejecute la cláusula. En este caso en particular, la variable anónima se utiliza como parte de una cláusula que existe para mostrar que siempre hay una ruta desde A hasta A

El Nuevo Mundo (III)

Concluimos nuestro proyecto ofreciendo la última parte del listado

El programa, aunque escrito para el Commodore 64, se puede ejecutar en ordenadores Amstrad con la introducción de unas pocas modificaciones menores. Todos los PRINT CHR$(147) se deben sustituir por CLS. Todas las líneas que aguarden una pulsación de tecla, como:

< n.º línea > GET I$:IF I$=" " THEN < n.º línea >

se deben reemplazar por:

< n.º línea > I$=" ": WHILE I$=" ":I$=INKEY$: WEND

Por último, para que se seleccione la modalidad de visualización en pantalla a 40 columnas se debe insertar la siguiente línea:

5 MODE 1

```
6530 REM BOTE SALVAVIDAS
6535 IF M(1)=1 THEN RETURN
6536 PRINTCHR$(147)
6540 M(1)=1
6550 S$="SE HA AVISTADO UN BOTE SALVAVIDAS*":GOSUB 9100
6552 S$="NAVEGANDO A LA DERIVA A LO LEJOS* ":GOSUB 9100
6554 PRINT:GOSUB 9200
6556 S$="A TRAVES DEL CATALEJO VES*":GOSUB 9100
6558 S$="QUE CONTIENE:*":GOSUB 9100
6560 PRINT:GOSUB 9200
6562 S$="4 PERSONAS*":GOSUB 9100
6563 GOSUB 9200
6564 S$="Y UN GRAN COFRE!*":GOSUB 9100
6566 PRINT:GOSUB 9200
6568 S$="SI ALTERAS TU RECORRIDO*":GOSUB 9100
6570 S$="PARA RECOGERLOS*":GOSUB 9100
6572 S$="TARDARAS DOS DIAS MAS*":GOSUB 9100
6574 PRINT:GOSUB 9200
6576 S$="QUIERES RESCATARLOS (S/N)*":GOSUB 9100
6578 INPUT I$:I$=LEFT$(I$,1)
6580 IF I$<>"S" AND I$<>"N" THEN 6578
6585 IF I$="S" THEN 6600
6588 PRINT:GOSUB 9200
6590 S$="EL BOTE SALVAVIDAS DESAPARECE....*":GOSUB 9100
6592 PRINT:GOSUB 9200
6594 S$=K$:GOSUB 9100
6596 GET I$:IF I$=" " THEN 6596
6599 RETURN
6600 PRINT:GOSUB 9200
6610 EW=EW+2/7
6625 IF CN <>16 THEN 6630
6627 S$="NO PUEDES RECOGERLOS*":GOSUB 9100
6628 S$="PORQUE NO TIENES LUGAR EN EL BARCO*":GOSUB
     9100
6629 GOTO 6592
6630 X=16-CN:IF X>3 THEN 6635
6632 S$="EN EL BARCO SOLO TIENES LUGAR PARA*":GOSUB
     9100
6633 PRINT X;"PERSONAS MAS"
6634 PRINT:GOSUB 9200
6635 S$="RECOGES:*":GOSUB 9100
6638 X=0
6640 FOR T=1 TO 16
6645 IF TS(T,1)<>0 THEN 6679
6650 X=X+1
6655 IF X>4 THEN T=16:GOTO 6679
6660 CN=CN+1
6665 TS(T,1)=INT(RND(1)*5)+1
6668 TS(T,2)=INT(RND(1)*50)+50
6670 PRINT"1 ";C$(TS(T,1))
6679 NEXT
6680 PRINT:GOSUB 9200
6682 S$="EL COFRE CONTIENE:*":GOSUB 9100
6685 FOR T=1 TO 4
6690 X=INT(RND(1)*10)+10
6692 PRINT X;U$(T);"S DE ":P$(T)
6693 IF PA(T)=-999 THEN PA(T)=0
6694 PA(T)=PA(T)+X
6695 NEXT
6699 GOTO 6592
```



Quizás una epidemia haga presa de la tripulación...

```
6700 REM AZOTE DE LA EPIDEMIA
6705 IF M(2)=1 THEN RETURN
6706 PRINTCHR$(147)
6710 M(2)=1
6712 S$="SE DECLARA UNA EPIDEMIA*":GOSUB 9100
6714 PRINT:GOSUB 9200
6716 X=1
6718 FOR T=1 TO 16
6720 IF TS(T,1)=2 AND TS(T,2)<>0 AND TS(T,2)<>-999 THEN X
     =0:T=16
6722 NEXT
6724 Y=1
6726 IF OA(1)<>0 AND OA(1)<>-999 THEN Y=0
6730 X=((X+Y)*10)+5
```

...y que los tripulantes puedan o no sobreponerse a la enfermedad dependerá de que usted haya contratado a un médico y comprado medicinas antes de zarpar

```
6732 I$="TIENES ":IF X=0 AND Y=0 THEN 6740
6734 IF X=1 THEN S$="SIN NINGUN MEDICO*":GOSUB 9100:I$"NI "
6736 IF Y=1 THEN S$=I$+"NINGUNA MEDICINA*":GOSUB 9100
6740 S$="MUCHOS DE LOS TRIPULANTES ESTAN AFECTADOS*":GOSUB 9100
6745 PRINT:GOSUB 9200
6750 X=0
6755 FOR T=1 TO 16
6756 IF RND(1) <.3 THEN 6775
6760 IF TS(T,2)=0 OR TS(T,2)=-999 THEN 6775
6765 TS(T,2)=TS(T,2)-Z
6770 IF TS(T,2)<1 THEN TS(T,2)=-999:X=X+1
6775 NEXT
6776 IF Y=1 THEN 6780
6777 S$="SE HAN UTILIZADO LA MITAD DE TUS MEDICINAS*":GOSUB 9100
6778 OA(1)=INT((OA(1)/2)+.5)
6780 PRINT:GOSUB 9200
6785 IF X=0 THEN 6797
6790 PRINT"Y";X;
6792 S$="TRIPULANTES MURIERON*"
6794 IF X=1 THEN S$="TRIPULANTE MURIO*"
6795 GOSUB 9100
6796 PRINT:GOSUB 9200
6797 S$=K$:GOSUB 9100
6798 GET I$:IF I$=" " THEN 6798
6799 RETURN
```

El barco puede ser atacado por piratas...

```
6800 REM PIRATAS
6805 IF M(3)=1 THEN RETURN
6810 X=0
6812 FOR T=1 TO 16
6814 IF TS(T,2)=0 OR TS(T,2)=-999 THEN X=X+1
6815 NEXT
6816 IF X=16 THEN RETURN
6818 M(3)=1
6820 PRINT CHR$(147)
```

...¿habrá a bordo algunas armas para defenderse de ellos y reducir las pérdidas al mínimo?

```
6822 S$=" EL BARCO ES ATACADO POR PIRATAS!*":GOSUB 9100
6824 PRINT;GOSUB 9200
6825 K=2
6826 IF OA(2)=0 OR OA(2)=-999 THEN K=4
6828 S$=" A PESAR DE TUS ARMAS*"
6830 IF K=4 THEN S$="NO TIENES ARMAS*"
6832 GOSUB 9100
6835 X=0
6836 FOR T=1 TO 16
6838 IF TS(T,2)=0 OR TS(T,2)=-999 THEN 6845
```



```
6840 X=X+1:TS(T,2)=-999
6842 IF X=K THEN T=16
6845 NEXT
6850 PRINT X;
6855 S$="TRIPULANTE HA RESULTADO MUERTO*"
6856 IF X>1 THEN S$="TRIPULANTES HAN RESULTADO MUERTOS*"
6860 GOSUB 9100
6865 PRINT:GOSUB 9200
6890 S$=K$:GOSUB 9100
6895 GET I$:IF I$=" " THEN 6895
6899 RETURN
6900 REM TIMON
6905 IF M(4)=1 THEN RETURN
6910 PRINT CHR$(147)
6915 M(4)=1
6920 S$="HAY PROBLEMAS CON EL TIMON!*":GOSUB 9100
6925 PRINT:GOSUB 9200
6928 X=4
6930 FOR T=1 TO 16
6935 IFTS(T,1)=3 AND TS(T,2)<>0 AND TS(T,2)<>-999 THEN X=1:T=16
6938 NEXT
6940 S$="A PESAR DE QUE TIENES UN MECANICO*"
6945 IF X=4 THEN S$="NO TIENES MECANICO Y*"
6950 GOSUB 9100
6955 S$=TU VIAJE DURARA*":GOSUB 9100
6960 PRINT X;"SEMANAS MAS"
6965 EW=EW+X
6967 PRINT:GOSUB 9200
6969 S$=K$:GOSUB 9100
6970 GET I$:IF I$=" " THEN 6970
6975 RETURN
7000 REM TORMENTA
7005 IF M(5)=1 THEN RETURN
7010 PRINTCHR$(147)
7015 M(5)=1
7020 S$="EL VIENTO TE APARTA DE TU RUMBO*":GOSUB 9100
7022 S$="DURANTE UNA TORMENTA!*:GOSUB 9100
7025 PRINT:GOSUB 9200
7028 X=2
7030 FOR T=1 TO 16
7035 IF TS(T,1)=4 AND TS(T,2)<>0 AND TS(T,2)<>-999 THEN X=1:T=16
7038 NEXT
7040 S$=A PESAR DE QUE CUENTAS CON UN OFICIAL*"
7045 IF X=2 THEN S$="NO TIENES NINGUN OFICIAL Y*"
7049 GOTO 6950
```

Si se te están acabando las provisiones, quizá quieras poner rumbo hacia esta isla y reabastecerte, pero al hacerlo tal vez el viaje se prolongue

```
7050 REM ISLA
7055 IF M(6)=1 THEN RETURN
7060 PRINTCHR$(147)
7065 M(6)=1
7070 S$="TUS CARTAS INDICAN LA EXISTENCIA DE UNA ISLA*":GOSUB
     9100
7071 S$="EN LA QUE PODRIAS*":GOSUB 9100
7072 S$="REABASTECERTE DE PROVISIONES*":GOSUB 9100
7073 S$="PERO SI TE DESVIAS HACIA ELLA*":GOSUB 9100
7074 S$="SUPONDRA UNA MAYOR DURACION DEL VIAJE*":GOSUB
     9100
7075 PRINT:GOSUB 9200
7080 S$="QUIERES IR A LA ISLA*":GOSUB 9100
7082 INPUT I$:I$=LEFT$(I$,1)
7084 IF I$<>"S" AND I$<>"N" THEN 7082
7086 PRINT:GOSUB 9200
7090 IF I$="N" THEN 7145
7100 S$="LLEGAS A LA ISLA*":GOSUB 9100
7105 S$="Y CONSIGUES:*":GOSUB 9100
7106 IF B$="N" THEN 7110
7107 PRINT:GOSUB 9200
7108 PRINT"NADA!":GOSUB 9200
7109 S$="(RECUERDA EL ALBATROS!)":GOSUB 9100;GOTO 7130
7110 FOR T=1 TO 4
7112 IF RND(1)<.25 THEN 7129
7115 X=INT(RND(1)*10)+5
7120 PRINT X;U$(T);"S DE";P$(T)
7122 IF PA(T)=-999 THEN PA(T)=0
7125 PA(T)=PA(T)+X
7129 NEXT
7130 S$="PERO AHORA EL VIAJE DURARA*":GOSUB 9100
7135 X=INT(RND(1)*2)+1
7139 PRINT X;:S$="SEMANAS MAS*":GOSUB 9100
7140 EW=EW+X
7145 PRINT:GOSUB 9200
7150 S$=K$:GOSUB 9100
7155 GET I$:IF I$=" " THEN 7155
7159 RETURN
```

Siete factores se pueden combinar para favorecer la rebelión de la tripulación. Esta rutina genera un factor de amotinamiento, MF, comprobando las condiciones que prevalecen al final de cada semana del viaje y sumándole a MF las estimaciones apropiadas

```
7200 REM MOTIN
7210 MF=0
7215 IF M$="S" THEN MF=MF+30
7220 NC=0
7225 FOR T=1 TO 16
7228 IF TS(T,1)=5 AND TS(T,2)<>0 AND TS(T,2)<>-999 THEN NC=1:T=16
7230 NEXT
7235 IF NC=0 THEN MF=MF+30
7240 IF A$="S" THEN MF=MF-20
7245 IF B$="S" THEN MF=MF+30
7250 IF CN>12 THEN MF=MF+30
7255 IF WT>MO THEN MF=MF+30
7260 IF WK>8 THEN MF=MF+((WK-8)*10)
7275 MF=MF+INT(RND(1)*30)
```

Si el factor de amotinamiento es mayor que 75 (pero menor que 100), se le advierte al jugador del descontento de la tripulación

```
7280 IF MF<75 THEN RETURN
7282 PRINT CHR$(147)
7284 IF MF>100 THEN 7300
7285 S$="LA SITUACION EN EL BARCO*":GOSUB 9100
7286 S$="ESTA EMPEORANDO*":GOSUB 9100
7287 S$="Y ALGUNOS DE LOS TRIPULANTES*":GOSUB 9100
7288 S$="HABLAN YA DE MOTIN!*":GOSUB 9100
7290 PRINT:GOSUB 9200
7292 S$=K$:GOSUB 9100
7294 GET I$:IF I$=" " THEN 7294
7299 RETURN
```

Si el factor de amotinamiento se eleva por encima de 100, la tripulación se subleva, listándose sus quejas antes de dejar al capitán en un bote a la deriva

```
7300 PRINT CHR$(147)
7305 PRINT:GOSUB 9200
7310 S$="LA TRIPULACION SE HA AMOTINADO*":GOSUB 9100
7312 S$="PORQUE:*":GOSUB 9100
7313 X=0
7314 IF M$<>"S" THEN 7320
7315 GOSUB 9200:X=X+1:PRINT X;
7316 S$="HAN ESTADO A MEDIA RACION*":GOSUB 9100
7318 S$=" DURANTE PARTE DEL VIAJE*":GOSUB 9100
7320 IF NC<>0 THEN 7325
7321 GOSUB 9200:X=X+1:PRINT X;
7322 S$="NO HAY COCINERO*":GOSUB 9100
7324 S$=" Y LA COMIDA ES ASQUEROSA*":GOSUB 9100
7325 IF B$<>"S" THEN 7330
7326 GOSUB 9200:X=X+1:PRINT X;
7327 S$="EL ALBATROS FUE ABATIDO!*":GOSUB 9100
7330 IF CN<13 THEN 7335
7331 GOSUB 9200:X=X+1:PRINT X;
7332 S$="LA TRIPULACION ESTA HACINADA*":GOSUB 9100
7335 IF MO>=WT THEN 7340
7336 GOSUB 9200:X=X+1:PRINT X;
7337 S$="NO HAY SUFICIENTE ORO*":GOSUB 9100
7338 S$=" PARA PAGARLES SUS SALARIOS*":GOSUB 9100
7340 IF WK<=8 THEN 7350
7341 GOSUB 9200:X=X+1:PRINT X;
7342 S$="LLEVAN NAVEGANDO*":GOSUB 9100
7343 S$=" MAS DE 8 SEMANAS*":GOSUB 9100
7350 PRINT:GOSUB 9200
7360 S$="LA TRIPULACION SE APODERA DEL BARCO*":GOSUB 9100
7362 GOSUB 9200
7363 S$="Y EMPRENDE EL REGRESO*":GOSUB 9100
7370 GOSUB 9200
7372 S$="DEJANDOTE A TI ABANDONADO*":GOSUB 9100
7373 S$="EN UN BOTE A LA DERIVA*":GOSUB 9100
7374 S$="OJALA QUE ALGUIEN TE RECOJA*":GOSUB 9100
7375 PRINT:GOSUB 9200
7380 PRINT"      FIN DEL JUEGO"
7382 END
```

Estas cortas rutinas se utilizan mucho en el programa para producir efectos tales como retardar la salida por pantalla

```
9100 REM IMPRESION LENTA DE S$
9110 FOR S3=1 TO 32
9115 IF MID$(S$,S3,1)="*" THEN S3=32:GOTO 9140
9120 PRINT MID$(S$,S3,1);
9130 FOR S4=1 TO 25:NEXT
9140 NEXT:PRINT
9199 RETURN
9200 REM BUCLE DE DEMORA
9210 FOR S5=1 TO 1000:NEXT
9299 RETURN
9300 REM REDUCIR FORTALEZA TRIPULACION EN FUNCION DE WF
9310 FOR S1=1 TO 16
9315 IF TS(S1,2)=0 THEN 9340
9320 TS(S1,2)=TS(S1,2)-WF
9330 IF TS(S1,2)<1 THEN TS(S1,2)=-999
9340 NEXT
9399 RETURN
```



El barco llega a su destino...

```
10000 REM LLEGADA AL NUEVO MUNDO
10001 PRINTCHR$(147):GOSUB 9200
10005 S$="LLEGAS AL NUEVO MUNDO*":GOSUB 9100
10006 PRINT:GOSUB 9200
10007 S$="MIENTRAS TE APROXIMAS A LA COSTA*":GOSUB 9100
10009 S$="SALEN LOS NATIVOS EN CANOAS PARA RECIBIRTE*":GOSUB 9100
10010 PRINT:GOSUB 9200
10015 IF OA(2)=0 THEN 10050
10017 S$="SU ASPECTO ES FIERO Y ESTAN ARMADOS!!*":GOSUB 9100
10018 PRINT:GOSUB 9200
10020 S$="ABRES FUEGO? (S/N)*":GOSUB 9100
10022 INPUT I$:I$=LEFT$(I$,1)
10024 IF I$<>"N" AND I$<>"S" THEN 10022
10026 IF I$="N" THEN 10050
10028 PRINT:GOSUB 9200
10030 S$="HAN MUERTO MUCHOS NATIVOS*":GOSUB 9100
10032 S$="PERO DURANTE LA NOCHE*":GOSUB 9100
10034 S$="OTROS REGRESAN*":GOSUB 9100
10036 S$="Y LE PRENDEN FUEGO A TU BARCO!!!!*":GOSUB 9100
10038 PRINT:PRINT:GOSUB 9200
10040 S$="      JUEGO TERMINADO*":GOSUB 9100
10042 END
10044 GOTO 10042
10050 S$="TE LLEVAN A CONOCER A SU*":GOSUB 9100
10052 S$="JEFE. YA HA CONOCIDO ANTES A GENTE DE*":GOSUB 9100
10054 S$="TU RAZA Y SE MUESTRA MUY AMABLE*":GOSUB 9100
10056 GOSTUB 9200
10058 S$="LA TRIPULACION HA COMIDO Y ESTA DESCANSANDO*":
      GOSUB 9100
10060 PRINT:GOSUB 9200
10062 S$="MAÑANA COMENZARA LA ACTIVIDAD COMERCIAL*":GOSUB 9100
10064 PRINT:GOSUB 9200
10066 S$=K$:GOSUB 9100
10068 GET I$:IF I$=" " THEN 10068
10069 RETURN
```

...y comienza la actividad comercial

```
10070 PRINTCHR$(147):GOSUB 9200:REM INTERCAMBIO
10072 IF OA(2)=0 THEN 10080
10074 S$="EL JEFE NO QUIERE TUS ESCOPETAS*":GOSUB 9100
10076 S$="PORQUE PODRIAN ACARREARLE PROBLEMAS*":GOSUB 9100
10078 PRINT:GOSUB 9200
10080 IFOA(3)<>0 OR OA(4)<> 0 OR OA(5)<>0 OR OA(6)<>0 THEN 101000
10085 S$="NO TE QUEDA NINGUNA MERCANCIA*":GOSUB 9100
10090 S$="CON LA QUE COMERCIAR*":GOSUB 9100
10095 GOTO 10038
10100 S$="EN TRUEQUE POR LOS CUCHILLOS*":GOSUB 9100
10102 S$="SAL TELA O JOYAS QUE POSEAS*":GOSUB 9100
10104 S$="TE OFRECE PERLAS, FIGURILLAS*":GOSUB 9100
10106 S$="Y ESPECIAS*":GOSUB 9100
10108 PRINT:GOSUB 9200
10110 S$="CUANDO SALISTE DE PUERTO ESTAS*":GOSUB 9100
10112 S$="VALIAN*":GOSUB 9100
10114 S$="PERLAS - 2 P DE ORO CADA UNA*":GOSUB 9100
10116 S$="FIGURILLAS - 2 P DE ORO CADA UNA*":GOSUB 9100
10118 S$="ESPECIAS - 1 PIEZA DE ORO EL GRAMO*":GOSUB 9100
10120 PRINT:GOSUB 9200
10122 S$="PERO QUIZAS CUANDO VUELVAS A CASA*":GOSUB 9100
10124 S$="QUIZAS ESTOS VALORES HAYAN CAMBIADO*":GOSUB 9100
10125 PRINT:GOSUB 9200:S$=K$:GOSUB 9100
10126 GET I$:IF I$=" " THEN 10126
10130 FOR T=3 TO=6
10135 IF OA(T)=0 THEN 10200
10140 PRINTCHR$(147):GOSUB 9200
10145 PRINT"TIENES";OA(T);
10150 IF T=3 THEN S$="SACOS DE SAL*"
10151 IF T=4 THEN S$="BALAS DE TELA*"
10152 IF T=5 THEN S$="CUCHILLOS*"
10153 IF T=6 THEN S$="JOYAS*"
10155 GOSUB 9100
10156 PRINT:GOSUB 9200
10160 S$="A CAMBIO EL JEFE TE OFRECE*":GOSUB 9100
10165 PRINT"YA SEA";OA(T)*EQ(T-2,1);"PERLAS"
10166 PRINT"O BIEN";OA(T)*EQ(T-2,2);"FIGURILLAS"
10167 PRINT"     O";OA(T)*EQ(T-2,3);"GRAMOS DE ESPECIAS"
10168 PRINT:GOSUB 9200
10170 S$="QUIERES PERLAS, FIGURILLAS*":GOSUB 9100
10172 S$="O ESPECIAS?*":GOSUB 9100
10174 S$="(ENTRA 1, 2 o 3)*":GOSUB 9100
10175 INPUT I$
10176 I=VAL(I$):IF 1<1 OR I>3 THEN 10174
10180 AO(I(=AO(I)+(OA(T)*EQ(T-2,I))
10190 PRINT:PRINT"LAS ";T$(I);" SE CARGAN EN EL BARCO"
10192 S$=K$: GOSUB 9100
10194 GET I$:IF I$=" " THEN 10194
10200 NEXT
10210 PRINT:PRINT:GOSUB 9200
10215 S$="   FIN DEL INTERCAMBIO*":GOSUB 9100
10216 PRINT:GOSUB 9200
10218 S$="HAS OBTENIDO:*":GOSUB 9100
10220 PRINT AO(1);"PERLAS"
10222 PRINT AO(2);"FIGURILLAS"
10224 PRINT AO(3);"GRAMOS DE ESPECIAS"
10226 PRINT:GOSUB 9200
10228 S$=K$:GOSUB 9100
10229 GET I$:IF I$=" " THEN 10229
10230 RETURN
```



¿Quieres participar en una revuelta local? Las recompensas son elevadas... pero también lo son los riesgos si el golpe fracasa

```
10300 REM REVOLUCION
10305 IF OA(2)=0 THEN RETURN
10310 PRINTCHR$(147):GOSUB 9200
10315 S$="DURANTE LA NOCHE UN RIVAL DEL*":GOSUB 9100
10316 S$="JEFE VISITA EL BARCO EN SECRETO*":GOSUB 9100
10317 PRINT:GOSUB 9200
10318 S$="QUIERE COMPRAR TUS ESCOPETAS*":GOSUB 9100
10320 S$="PARA UNA REVOLUCION*":GOSUB 9100
10322 PRINT:GOSUB 9200
10324 S$="TE OFRECE 30 PERLAS POR CADA ESCOPETA*":GOSUB 9100
10326 S$="LE VENDES LAS ESCOPETAS? (S/N)*":GOSUB 9100
10328 INPUT I$:I$=LEFT$(I$,1)
10330 IF I$<>"N" AND I$<>"S" THEN 10328
10332 IF I$="S" THEN 10400
10334 PRINT:GOSUB 9200
10336 S$="EL JEFE SE ENTERA Y SE SIENTE*":GOSUB 9100
10338 S$="AGRADECIDO HACIA TI*":GOSUB 9100
10340 S$="TE DA PROVISIONES GRATIS*":GOSUB 9100
10342 S$="PARA EL VIAJE DE REGRESO*":GOSUB 9100
10344 GOSUB 9200
10345 IF RND(1)<.75 THEN 10350
10346 S$="Y 50 PERLAS!!*":GOSUB 9100
10348 AO(1)=AO(1)+50
10350 PRINT:GOSUB 9200
10352 S$=K$:GOSUB 9100
10354 GET I$:IF I$=" " THEN 10354
10359 RETURN
10400 PRINTCHR$(147):GOSUB 9200
10405 IF RND(1)<.75 THEN 10450
10410 S$="LA REVOLUCION HA TRIUNFADO*":GOSUB 9100
10412 PRINT:GOSUB 9200
10415 S$="EL NUEVO JEFE TE RECOMPENSA CON*":GOSUB 9100
10420 S$="PROVISIONES GRATUITAS PARA EL VIAJE*":GOSUB 9100
10425 S$="DE REGRESO.*":GOSUB 9100
10429 AO(1)=AO(1)+(OA(2)*30):REM SUMAR PERLAS
10430 OA(2)=0
10431 GOTO 10350
10450 S$="LA REVOLUCION FRACASA!!*":GOSUB 9100
10452 PRINT:GOSUB 9200
10455 S$="EL VIEJO JEFE ESTA ENFADADO CONTIGO*":GOSUB 9100
10457 LE PRENDE FUEGO A TU BARCO Y ROBA*":GOSUB 9100
10458 S$="TODO!!*":GOSUB 9100
10459 PRINT:GOSUB 9200
10460 S$="      JUEGO TERMINADO!!*":GOSUB 9100
10462 END
10464 GOTO 10462
10500 REM FIN DEL VIAJE
```

El barco regresa y se realiza una evaluación de la actuación del jugador como comerciante

```
10501 PRINTCHR$(147):GOSUB 9200
10505 S$="CON UNA TRIPULACION FUERTE Y VIENTOS*":GOSUB 9100
10507 S$="FAVORABLES EL VIAJE DE REGRESO VA*":GOSUB 9100
10508 S$="BIEN Y DURA SOLO 8 SEMANAS*":GOSUB 9100
10512 WW=0
10514 FOR T=1 TO 5
10516 WW=WW+(8*CC(T)*WG(T))
10518 NEXT
10519 PRINT:GOSUB 9200
10520 S$="FACTURA SALARIAL PARA EL VIAJE DE REGRESO*":GOSUB
      9100
10522 PRINT WW;"PIEZAS DE ORO"
10524 PRINT:GOSUB 9200
10526 S$="CUANDO REGRESAS*":GOSUB 9100
10528 S$="PARA VENDER TUS MERCANCIAS*":GOSUB 9100
10530 S$="ESTAS VALEN ENTONCES*":GOSUB 9100
10532 PRINT"PERLAS - ";V2(1);"PIEZAS DE ORO"
10534 PRINT"FIGURILLAS - ";V2(2);"PIEZAS DE ORO"
10536 PRINT"ESPECIAS - ";V2(3);"PIEZAS DE ORO"
10538 PRINT:S$="OBTIENES UN TOTAL DE*":GOSUB 9100
10540 X=(AO(1)*V2(1))+(AO(2)*V2(2))+(AO(3)*V2(3))
10542 PRINT X:"PIEZAS DE ORO"
10545 PRINT:S$=K$:GOSUB 9100:PRINT
10547 GET I$:IF I$=" " THEN 10547
10550 S$="AHORA POSEES*":GOSUB 9100
10552 PRINT MO+X;"PIEZAS DE ORO"
10555 PRINT:GOSUB 9200
10556 S$="LA FACTURA SALARIAL PARA EL VIAJE ES DE*":GOSUB 9100
10557 PRINT WT+WW;"PIEZAS DE ORO"
10559 PRINT:GOSUB 9200
10560 S$="TERMINAS EL VIAJE CON:*":GOSUB 9100
10562 Z=MO+X-WT-WW
10565 PRINT Z;"PIEZAS DE ORO"
10566 PRINT:GOSUB 9200
10567 PRINT:S$="TU CLASIFICACION ES:*":GOSUB 9100:PRINT
10568 IF Z>3200 THEN S$="CAPITALISTA DE PRIMER ORDEN*":GOSUB
      9100:END
10569 IF Z>2500 THEN S$="INSIGNE COMERCIANTE*":GOSUB 9100:END
10570 IF Z>2000 THEN S$="MERCACHIFLE DE III CLASE*":GOSUB
      9100:END
10571 IF Z>1000 THEN S$="MAS BIEN UN PRIMO*":GOSUB 9100:END
10572 S$="MAS PATO QUE PIRATA*"
10573 GOSUB 9100:END
```

Metro de sastre

Analicemos las rutinas ROM del OS del Spectrum que se encargan del sistema de archivo en cinta

Un Spectrum de Sinclair normal está provisto de sólo una interface para cinta que permite guardar y cargar programas y datos. Pero la adición de la Interface 1 da acceso a los microdrives, la interface serial y las redes de área local. Esto se considera como sistemas alternativos de archivo un poco como el BBC Micro posee sus propias alternativas. Sin embargo, en el BBC Micro se selecciona el sistema de archivo por medio de una instrucción *, mientras que en el Spectrum empleamos una sintaxis diferente de instrucciones para diferenciar las referidas al sistema de archivo en cinta de otras pertenecientes a otros sistemas de archivo disponibles para este ordenador. Por ejemplo, la instrucción

```
SAVE "hoja1" LINE 1
```

guardará un programa en BASIC en la cinta de modo que iniciará la ejecución desde la línea 1 cuando sea vuelto a cargar en el Spectrum. Para guardar un programa de modo semejante pero en un sistema de archivo microdrive es necesaria la siguiente instrucción increíblemente alambicada:

```
SAVE * "m";1;"hoja1" LINE 1
```

La "m" de la instrucción especifica el microdrive que se emplea; el 1 que sigue indica el número de unidad de disco solicitado; y el nombre del archivo (en este caso "hoja1") viene después. Ya veremos cómo funciona este sistema de archivo en microdrive. Veamos ahora el empleo del sistema de archivo en cinta a partir de programas en código máquina.

Independientemente de los datos que se guardan en la cassette, éstos se organizan de la manera que ilustra el dibujo. Los primeros 19 bytes enviados a la cinta son considerados como el *encabezamiento* (*header*), y contienen datos relativos al siguiente bloque de datos en la cinta. Los bytes primero y último del encabezamiento son dados por las rutinas del sistema operativo que escriben los bytes de datos en la cinta. Los otros deben ser codificados antes de ser llamada la rutina ROM que escribe los datos en la cinta. El byte "tipo" indica al OS, en la carga, la naturaleza de los datos que están en el bloque. Después está el nombre del archivo en 10 bytes (lo que explica por qué los nombres de los archivos en el Spectrum tienen un máximo de 10 caracteres). Si el archivo tiene un nombre de menos de 10 caracteres, los espacios sobrantes son rellenados con el código ASCII de espacio (32). Los datos del encabezamiento desde el byte 11 en adelante proporcionan otros detalles al OS, tales como la información referente al lugar donde se han de cargar los datos en la memoria. El byte final del

00 — Indica un ENCABEZAMIENTO
0 type — 0 para programa en BASIC; 1 para tablas de números; 2 para tabla de series; 3 para archivo CODIGO (CODE)
1 — 10 bytes para nombre fichero
11 lo-byte, hi-byte — Longitud del bloque de datos que sigue
13 lo-byte, hi-byte — En programa BASIC, contiene número línea para inicio de ejecución del programa o la dirección de inicio del código a guardar
15 lo-byte, hi-byte — Longitud del programa, si es en BASIC
17 parity — Empleado por el OS para comprobación de los datos
silence
255 — Indica un bloque de datos
Aquí están los datos a guardar

Mucho carrete
El sistema de archivo en cinta del Spectrum guarda los datos en dos fases. Primero da salida a un bloque de ENCABEZAMIENTO que contiene la información sobre el bloque de DATOS que le sigue. El formato del bloque de ENCABEZAMIENTO es siempre el mismo, independientemente de la longitud de lo que se contiene en el bloque de DATOS que le sigue. Obsérvese que el byte final de paridad de este bloque es generado por el OS según el contenido de los bytes que le preceden, obrando así como una comprobación en el momento de la carga (LOAD)

encabezamiento se usa para comprobación de errores cuando ya se ha leído la cinta: si se detecta alguno, se genera el mensaje de error correspondiente.

El bloque principal de los datos comienza con un byte único que retiene el valor 255; el cual, una vez más, es dado por la rutina ROM de "escritura en cinta". Siguen después los datos, detrás de los cuales viene otro byte de comprobación, igualmente generado por la rutina citada. Tanto en el encabezamiento como en los bloques de datos, el valor efectivo escrito en el último byte del bloque depende de los enviados inmediatamente antes del citado byte.

Veamos ahora cómo pueden emplearse las rutinas de la interface para cassette a partir de programas escritos en código máquina. Las rutinas ROM relacionadas con operaciones en cassette se encuentran, en el Spectrum, en las direcciones que van de la &04C2 y &09F3 de la ROM del ordenador. Es, por tanto, una pieza de software dimensionable. Lo cual no debe sorprendernos ya que todas las operaciones temporizadoras y toda generación de sonidos que exigen información colocada en cinta se basan en software creado para el Sinclar Spectrum.

No existen variables de sistema específicas para operaciones en cinta, pero el siguiente cuadro contiene unas cuantas que "incidentalmente" son empleadas por las rutinas.

Variable del sistema	Descripción
TADDR (en posiciones 23668 y 23669)	Empleada por ruts. ROM para distinguir qué opers., en cinta se han de ejecutar al interpretar la instrucción en BASIC
BORDCR (en 23624)	Usada para restaurar el color del recuadro (Border) según su color inicial, tras una operación en cinta
XPTR (en 23647 y 23648)	Usada para almacenar temporalmente el registro IX

Las variables de sistema VAR, ELINE y PROG también se emplean en la carga de información desde la cinta al ordenador.

Almacenamiento en cinta

Examinemos ante todo esta operación, cuya rutina se encuentra en la dirección de la ROM &04C2. En su empleo normal, esta rutina es llamada *dos veces*; una vez para guardar la información de encabezamiento y la segunda para almacenar los datos efectivos que deseamos guardar. Los requisitos de entrada para esta rutina se dan en el cuadro inferior de la columna precedente.

Registro	Encabez.	Bloque de datos
A	0	255
DE	17	Longitud datos
IX	Dirección de inicio del encabezamiento o de inicio de los datos	

Como se deduce de este cuadro, el registro IX se usa para apuntar los datos que hay que guardar. Para clarificar esto, veamos un ejemplo sencillo de cómo se guarda un bloque de datos en la cinta. El siguiente programa guardará 100 bytes de datos (comenzando en la dirección 0000 de la ROM) en cinta.

```
                 ;rutina para guardar 100 bytes, comenzando en
                 ;la dirección 0000, en cinta
                 ;envío encabezamiento
3E00                      ld    a,0          ;indica un BLOQUE ENCABEZ.
DDE5                      push  ix           ;guarda ix en pila
DD21D328                  ld    ix,header    ;suma del bloque encabez.
111100                    ld    de,17        ;núm. de bytes encabez.
CDC204                    call  #04c2        ;escribe en cinta encabez.
                 ;envío datos
DD210000                  ld    ix,0000      ;suma del 1.er byte a guardar
116400                    ld    de,100       ;núm. bytes a guardar
3EFF                      ld    a,255        ;indica un BLOQUE DATOS
CDC204                    call  #04c2        ;escribe datos en cinta
DDE1                      pop   ix           ;restaura valor de IX
C9                        ret                ;vuelta al BASIC
03               header:  defb  3            ;tipo datos 3=CODIGO bloque
54455354                  defm  "TESTPROG    ;f/nombre+espacios=10 caracteres
64                        defb  100          ;byte-lo de longitud datos
00                        defb  0            ;byte-hi de longitud datos
00                        defb  00           ;byte-lo de dir. inicio
00                        defb  00           ;byte-hi de dir. inicio
00                        defb  00           ;empleado sólo para BASIC
00                        defb  00           ;para número línea inicio
```

Llamando a esta rutina nos ahorramos una parte importante de memoria. No obstante, no se emitirá ninguna indicación o mensaje al contrario que en las rutinas de gestión de cassette desde el BASIC. Pero esto no es problema ya que los datos en la cinta pueden volverse a cargar para ver si el programa ha realizado su tarea correctamente. Esto se hace con la instrucción:

```
LOAD "TESTPROG" CODE 40000
```

Ahora compárense los bytes cargados con los bytes retenidos en las posiciones entre la 0 y la 99 de la ROM. Recuérdese que aunque el bloque de datos ENCABEZAMIENTO comienza con el byte tipo, el primer byte efectivamente enviado a la cinta viene dado por el OS (0 para el encabezamiento y 255 para el bloque de datos).

Una variante útil de este programa puede permitirnos guardar un programa en BASIC desde dentro de un listado en código máquina. El bloque ENCABEZAMIENTO al final de nuestro anterior listado será así alterado para leer:

```
00          header:   defb 0               ;0=programa BASIC
73737373              defm   "sssss        ;nombre fichero con 10 caract.
                      sssss"
00                    defb   nn            ;byte-lo longitud de
00                    defb   nn            ;byte-hi prog+variables
00                    defb   nn            ;byte-lo número línea
00                    defb   nn            ;byte-hi de inicio
00                    defb   nn            ;byte-lo sólo longitud de
00                    defb   nn            ;byte-hi programa
```

La longitud del programa y las variables pueden hallarse restando el valor contenido en PROG del valor contenido en ELINE, y la longitud del programa sólo puede obtenerse restando el valor de PROG del contenido en VARS. Si no se desea que el programa se ejecute una vez cargado, basta poner la entrada 'número de línea en el que el programa comenzará a ejecutarse' con valor 32768. Igualmente, podemos simular la instrucción SAVE SCREEN$ desde el códi-



go máquina especificando la dirección de inicio del código que ha de guardarse como &4000 y la longitud como &1B00.

Carga desde cinta

La rutina ROM que carga datos desde la cinta de cassette también tiene que ser llamada dos veces: una para el ENCABEZAMIENTO y otra para el bloque efectivo de datos que ha de cargarse. La rutina se halla en la dirección &0556 en la ROM y sus requisitos de entrada se presentan en la siguiente tabla (junto con los del código de verificación):

Registro	Carga	Verificación
Flag C	1	0
A	0: encabez.; 255: bloque datos	
IX	Apunta el lugar de memoria donde serán cargados los bytes	
DE	Número de bytes por cargar; D debe estar entre 0 y 254	

Necesitamos algo de espacio de trabajo para esta rutina: unos 34 bytes, o lo suficiente para acomodar dos bloques de ENCABEZAMIENTO. La razón de esto es inmediata. Para cargar un archivo dado en una dirección dentro de la RAM que especifiquemos, primero debemos establecer un segundo bloque de ENCABEZAMIENTO con los detalles del archivo que deseamos cargar. Después comparamos los datos de ENCABEZAMIENTO de los archivos en cinta con el ENCABEZAMIENTO que hemos establecido en la memoria para colocar el archivo que queremos.

Al llamar a la rutina (en &0556) el flag C debe establecerse como se indica en el cuadro. Si se está verificando un fragmento de código, debe colocarse en la dirección apuntada por el registro IX. Al abandonar la rutina ROM de 'carga desde cinta', el flag C indicará el resultado de la operación. Si está a uno, es que la carga se hizo bien. Pero si se trataba de cargar un bloque de ENCABEZAMIENTO, y lo primero que se encuentra en cinta es un bloque de datos, el flag C se pondrá a cero. (Todos los errores de carga desde cinta son gestionados por el OS.) Veamos ahora un ejemplo que carga un ENCABEZAMIENTO (HEADER) desde la cinta a la RAM.

```
            ;carga un encabezamiento de cinta a RAM
37                  scf                 ;pone arrastre 1=CARGA
3E00                ld      a,0         ;0 indica un encabez
DDE5                push    ix          ;guarda ix en pila
DD2148EE            ld      ix,61000    ;carga encabez en 61000
111100              ld      de,17       ;núm bytes en encabez
CD5605              call    #0556       ;lo hace
DDE1                pop     ix          ;restaura ix
C9                  ret
```

Una vez que se ha cargado ENCABEZAMIENTO lo podemos examinar. Podemos comparar el nombre del archivo con el requerido, asegurarnos que se trata del tipo correcto de fichero y comprobar la longitud del bloque de datos si queremos. El programa siguiente comprueba sólo el nombre, y si es correcto carga el bloque de datos después del ENCABEZAMIENTO en una dirección particular de la memoria.

```
            localización y carga TESTPROG
DDE5                push    ix          ;guarda IX en pila
37        loop      scf                 ;pone arrastre para CARGA
3E00                ld      a,0         ;0=Fichero BASIC
111100              ld      de,17       ;núm de byte en encabez
DD21CA2B            ld      ix,head2    ;carga encabez en head 2
CD5605              call    #0556
D27B2B              jp      np,loop     ;si no hay encabez, reintentar
060A                ld      b,10        ;núm de bytes en nombre fichero
11BA2B              ld      de,head+1   ;apunta al nombre de fichero
                                        deseado
21CB2B              ld      h1,head2+1  ;apunta HL al nombre fichero hallado
1A        name:     ld      a,(de)      ;toma caract. encabez en a
BE                  cp      (h1)        ;compara con encabez hallado
2007                jr      nz,no       ;no igual, salida bucle
13                  inc     de          ;toma carac. siguiente
23                  inc     hl
05                  dec     b           ;decrementa contador
20F7                jr      nz,name     ;igual, comprueba carácter sig.
1802                jr      ok          ;todos los caracteres ok
18DB      no:       jr      loop        ;comprobación fallida, prueba de
                                        nuevo
          ;encabez. correcto, preparación carga datos
DD21B92B  ok:       ld      ix,head     ;toma encabezamiento en ix
DD6E0D              ld      l,(ix+13)   ;toma dirección para
DD660E              ld      h,(ix+14)   ;carga datos
E5                  push    h1          ;lleva dirección a pila
DDE1                pop     ix          ;la toma en ix
3EFF                ld      a,255       ;indica carga datos
          ;la instrucción siguiente exige que el usuario entre la longitud de los datos
11AF2B              ld      de,nn       ;entrar longitud datos
37                  scf                 ;indica una CARGA
CD5605              call    #0556       ;la hace
DDE1                pop     ix          ;restaura ix
C9                  ret
          ;ahora siguen detalles del nombre fichero deseado
03        head:     defb 3              ;indica bloque CODIGO
54455354            defm "TESTPROG"     ;nombre fichero
00                  defb 0              ;empleo para longitud datos
00                  defb 0              ;byte-hi long datos
48                  defb 72             ;byte-lo carga
EE                  defb 238            ;dirección de 61000
00                  defb 0              ;sólo para prog. BASIC
00                  defb 0              ;ditto
          ;ahora sigue espacio para encabezamiento cargado y su comprobación
          head2:    defs 17
```

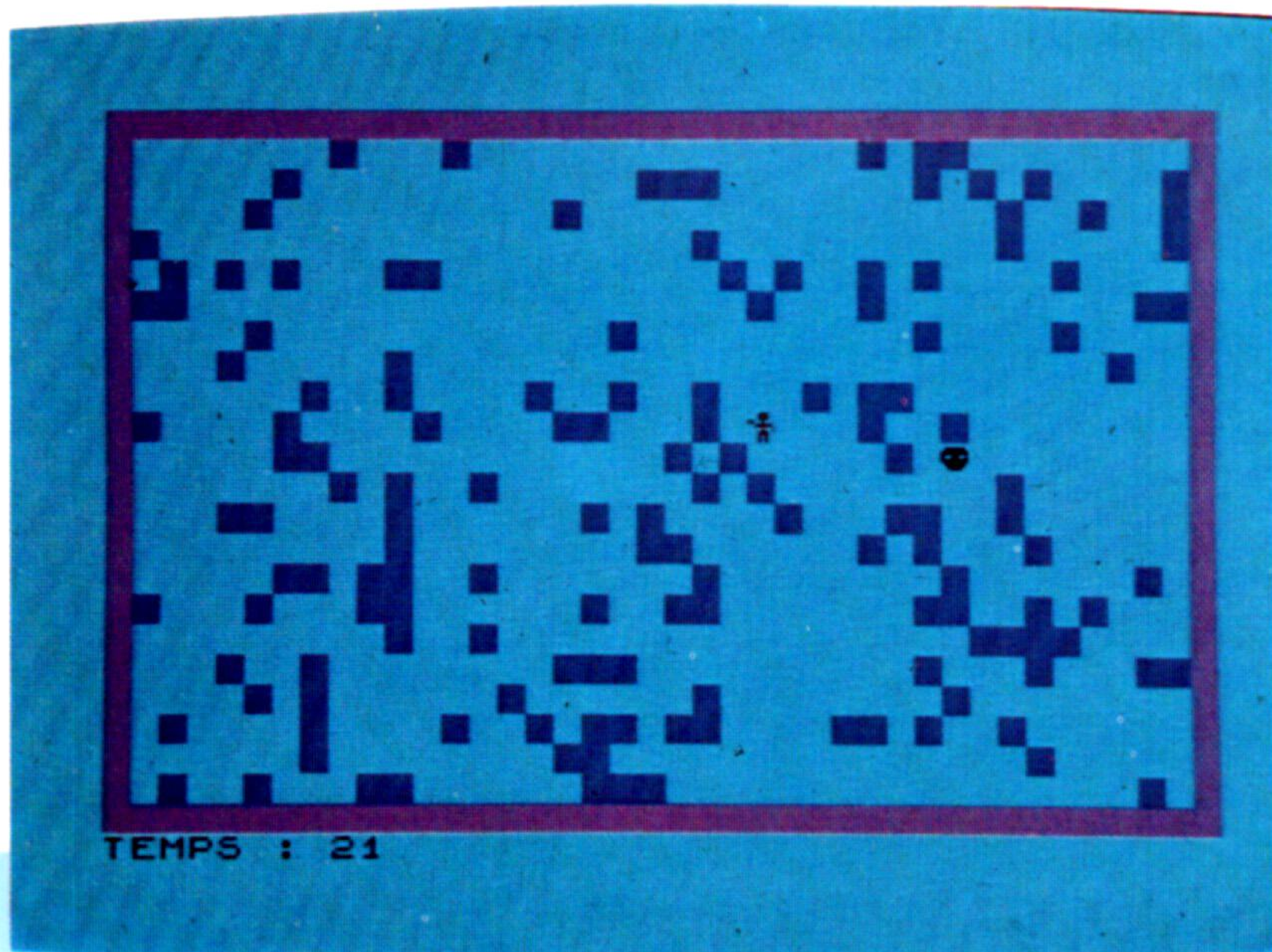
De nuevo nos encontramos con que no se envía mensaje alguno a la pantalla durante esta operación. La rutina cargará un bloque de datos llamado TESTPROG en la dirección especificada en el área de ENCABEZAMIENTO de la memoria. La rutina puede usarse para cargar archivos de nombre distinto, y se puede incluso añadir código máquina para comprobar que es correcto el tipo de archivo. Hay que observar que estas rutinas ROM, como las operaciones del BASIC SAVE y LOAD, pueden cancelarse pulsando la tecla Break.

Acabamos nuestro análisis de las operaciones de SAVE y LOAD del OS empleado por el Spectrum con un breve programa que lee encabezamientos de archivo desde cassette e imprime detalles sobre el fichero en pantalla, tales como su longitud, dirección de inicio, etc. La rutina en código máquina se almacena en la sentencia DATA, y carga sencillamente un bloque de ENCABEZAMIENTO como se ha señalado más arriba, examinando después el bloque desde el BASIC:

```
5 CLEAR 59999
10 FOR I=0 TO 19
20 READ A:POKE (60000+I) ,A
30 NEXT I
40 RANDOMIZE USR 60000
50 LET type=PEEK 60020
60 LET length=PEEK 60031+256*PEEK 60032
70 LET start=PEEK 60033+256*PEEK 60034
80 LET L$=" "
90 FOR I=60021 TO 60030: LET L$=L$+CHR$(PEEK I): NEXT I
100 PRINT "Nombre: ";L$
110 IF type=0 THEN LET f$="BASIC"
120 IF type=1 THEN LET f$="Tabla números"
130 IF type=2 THEN LET f$="Tabla serie"
140 IF type=3 THEN LET f$="CODIGO"
150 PRINT "Tipo: ";f$
160 IF type=0 THEN PRINT "Num. linea autoejecucion: ";start
170 IF type<>0 THEN PRINT "Direccion inicio: ";start
180 PRINT "Longitud: ";length
190 GOTO 40
200 DATA 221,229,62,0,55,221,33,116,234,17,17,0,205,86,5,48,241,221,225,201
```

Persecución

El ladrón ha escapado (está representado por una máscara negra). Usted tiene treinta minutos para encontrarlo y detenerlo con su ordenador Dragon



Atención, ¡no se precipite! Efectivamente, si se echa sobre el ladrón sin pensar, éste tiene todas las posibilidades de escapársele. La mejor manera de cogerlo es alcanzarlo de lado. (Es el método más eficaz a condición de no fallar.) Si no se siente lo bastante seguro de sí mismo, atáquelo de cara, lo cual es más fácil pero mucho menos eficaz, ya que no es tan discreto. Otro consejo: no intente perseguirlo; esto no le daría resultado, pues él es mucho más rápido que usted. Ha de observar sus movimientos, como un detective. Cuando le vea que da la vuelta, acérquese sin hacer ruido y sorpréndalo en el momento justo. Pero, recuerde, ¡el tiempo va pasando! Para desplazarse utilice la palanca de mando (*joystick*) o las teclas siguientes: <W>: arriba; <A>: izquierda; <S>: derecha; <Z>: abajo.

```
10 REM ****************
20 REM *  PERSECUCION  *
30 REM ****************
40 GOSUB 980
50 S=0
60 CN=159
70 CV=128
80 CP=191
90 GOSUB 640
100 ON JK GOTO 150
110 D$=INKEY$
120 D=(D$="A")-(D$="S")+32*((D$="W")-
    (D$="Z"))
130 IF D<>0 THEN DO=D
140 GOTO 230
150 K0=JOYSTK(0)
160 K1=JOYSTK(1)
170 IF K0<27 AND K1<27 THEN KS=-32
180 IF K0>36 AND K1<27 THEN KS=1
190 IF K0<27 AND K1>36 THEN KS=-1
200 IF K0>27 AND K1>27 THEN KS=32
210 IF KS<>0 THEN DO=KS
220 KS=0
230 T=T-0.1
240 PRINT@480,"TIEMPO :";INT(T+1);
250 IF T<0 THEN 400
260 P=P+DO
270 C=PEEK(P)
280 IF C=128 THEN 920
290 IF C<>159 THEN P=P1
300 POKE P1,CN
310 POKE P,CP
320 P1=P
330 V=V+DV
340 IF PEEK(V)<>CN THEN GOSUB 500
350 IF PEEK(V)<>CN THEN 330
360 POKE V1,CN
370 POKE V,CV
380 V1=V
390 GOTO 100
400 D$=INKEY$
410 IF R<S THEN R=S
420 PRINT@166,"TIEMPO TRANSCURRIDO";
430 PRINT@234,"PUNTUACION :";S;
440 PRINT@266,"RECORD :";R;
450 PRINT@326,"OTRA (S/N) ?";
460 D$=INKEY$
470 IF D$=" " THEN 460
480 IF D$<>"N" THEN 50
490 END
500 D2=D2+1
510 GOSUB 600
520 IF PEEK(V1+DV)=CN THEN
    V=V1+DV:RETURN
530 D2=D2-2
540 GOSUB 600
550 IF PEEK(V1+DV)=CN THEN
    V=V1+DV:RETURN
560 D2=D2-1
570 GOSUB 600
580 V=V1+DV
590 RETURN
600 IF D2>4 THEN D2=D2-4
610 IF D2<1THEN D2=D2+4
620 DV=(D2=1)-(D2=3)+32*((D2=2)-
    (D2=4))
630 RETURN
640 CLS 2
650 FOR I=1024 TO 1055
660 POKE I,175
670 POKE 448+I.175
680 NEXT I
690 FOR I=1 TO 13
700 POKE I*32+1024,175
710 POKE I*32+1055,175
720 NEXT I
730 FOR I=1 TO 70
740 GOSUB 890
750 POKE P,96
760 NEXT I
770 GOSUB 890
780 V=P
790 POKE V,CV
800 V1=V
810 GOSUB 890
820 POKE P,CP
830 P1=P
840 T=30
850 D0=0
860 DV=0
870 D2=0
880 RETURN
890 P=RND(414)+1056
900 IF PEEK(P)<>CN THEN 890
910 RETURN
920 FOR I=1 TO 5
930 SOUND 35,10
940 SOUND 5,10
950 NEXT I
960 S=S+1
970 GOTO 90
980 CLS
990 PRINT@200,"JOYSTICK (S/N)"
1000 D$=INKEY$
1010 IF D$=" " THEN 1000
1020 IF D$="S" THEN JK=1
1030 RETURN
```



mi compuTER
Editorial Delta, S.A.





10092

9 788485 822836